

Discos: Invesdisk 200

Desensamblador del Z-80

Proteccion del Software

QL Magazine
Hoja electrónica con ABACUS
AJEDREZ TRIDIMENSIONAL

¡¡MENUDO CAMBIO!!

Tráenos tu



SPECTRUM

y llévate un



SPECTRUM PLUS

Renuévate con INVESTRONICA.

Ahora INVESTRONICA te da la oportunidad de hacerte con el microordenador más moderno del mercado: EL SPECTRUM PLUS.
Sólo tendrás que entregarnos tu ZX SPECTRUM...
...lo demás será visto y no visto, el Spectrum Plus ya es tuyo.
Tener un ordenador Sinclair es la garantía de estar siempre a la última.

Apúntate a lo más nuevo.

El Spectrum Plus es lo más nuevo del mercado. Si tu Spectrum es estupendo; el Plus es fabuloso. Podrás disfrutar de un teclado profesional;
17 teclas más que el Spectrum, es decir 17 ventajas más... y por supuesto lo podrás utilizar con todos los programas y periféricos que ya tienes, puesto que **el SPECTRUM PLUS es totalmente compatible con todo el software y accesorios del spectrum.** Además INVESTRONICA, al realizar el cambio, **te da de nuevo 6 meses de garantía,** una nueva cassette de demostración y un libro de instrucciones a todo color.

No te lo pienses... cámbiate a lo último, tienes las de ganar.

Tenerlo, muy fácil

Manda tu ZX Spectrum (sin cables, ni fuente de alimentación) a tu Servicio Técnico Oficial (HISSA) más cercano, bien personalmente o por agencia de transportes (los gastos son por cuenta de INVESTRONICA) y en 48 horas ya podrás disfrutar de tu nuevo Spectrum Plus.
Sólo tienes que abonar (contra reembolso) 12.000 Pts. (*)



(*) 18.000 pts. si es de 16 K.

Dirígete a cualquiera de las delegaciones HISSA

C/. Aribau, n.º 80, Piso 5.º 1.º
Telfs. (93) 323 41 65 - 323 44 04
08036 BARCELONA

P.º de Ronda, n.º 82, 1.º E
Telf. (958) 26 15 94
18006 GRANADA

C/. San Sotero, n.º 3
Telfs. 754 31 97 - 754 32 34
28037 MADRID

C/. Avda. de la Libertad, n.º 6
bloque 1.º Entl. izq. D
Telf. (968) 23 18 34
30009 MURCIA

C/. 19 de Julio, n.º 10 - 2.º local 3
Telf. (985) 21 88 95
33002 OVIEDO

C/. Hermanos del Río Rodríguez, n.º 7 bis
Tel: (954) 36 17 08
41009 SEVILLA

C/. Universidad n.º 4 - 2.º 1.º
Telf. (96) 352 48 82
46002 VALENCIA

C/. Travesía de Vigo, n.º 32, 1.º
Telf. (986) 37 78 87
6 VIGO

Avda. de Gasteiz, n.º 19 A - 1.º D
Telf. (945) 22 52 05
01008 VITORIA

C/. Atares, n.º 4 - 5.º D
Telf. (976) 22 47 09
50003 ZARAGOZA

J M PUBLICIDAD



10

AÑO I

Director: Simeón Cruz









Los discos están de moda. Al análisis del Wafadrive del último número, le sigue ahora el Invesdisk. Coincidiendo con su comercialización, llega el Discovery, que presenta la novedad del disco de tres pulgadas. En fin, formatos para todos los gustos y precios prohibitivos igualmente para todos.

Nuestra preocupación educativa nos llevó este mes a Barcelona, donde entrevistamos a Damià García, gerente de una de las principales casas de software educativo. Estamos abiertos a cualquier duda o sugerencia que nos queráis plantear al respecto. Nuestro buzón está permanentemente abierto.

Finalmente recomendar efusivamente la lectura de la QL-sección. Enfrentamos al QL con el Macintosh a una batalla de Ajedrez.

Hasta el próximo número.

UNIDADES DE DISCO

INVESDISK 200

Nacido en Portugal e importado de Inglaterra, el Invesdisk 200 se presenta ahora en España como una unidad de discos seria para Spectrum, distribuida y avalada por Investrónica. Sin embargo, las diferencias con los restantes sistemas de almacenamiento no parecen ser tan decisivas.

La unidad de *diskette* de **Timex,** comercializada en España por **Investrónica** bajo el nombre de Invesdisk, ha sido uno de los últimos periféricos en llegar al mercado, y es el primer disco que utiliza el formato de tres pulgadas (no confundir con los *diskettes* de 3″1/2 de Sony), empleado por el Einstein y el Oric.

Este tipo de *floppys* permite su utilización por las dos caras, con lo que cada disco admite hasta 320 K. De las 160 K de cada cara, 16 K contienen el sistema operativo y 4 el directorio, por lo que restan 140 K libres para el usuario. La unidad incluye dos conexiones RS232C para comunicaciones o control de impresora, y un operativo con una gestión de archivos muy versátil y que utiliza sólo dos *bytes* (de una variable del sistema no usada) de la RAM del Spectrum.

Una caja incluye el controlador, con un *interface* para Spectrum. El modelo «manda» hasta cuatro unidades de *diskette,* e incluye los dos *ports* RS232C en su parte trasera.

En un módulo aparte se presenta la unidad de *diskettes,* y una tercera caja incluye la fuente de alimentación, que resulta suficiente para dos unidades de disco y el controlador. Si se quiere utilizar con tres o cuatro unidades, será necesaria una segunda fuente de alimentación.



El controlador trabaja en paralelo al Spectrum, con un procesador Z80A, memoria y dispositivos de entrada/salida propios. El Spectrum se limita a pasar los comandos al controlador, recibiendo el resultado. Viene dotado de un botón de RESET, que permite inicializar el sistema de disco sin afectar al Spectrum. Asimismo, el *interface* de conexión con el Spectrum viene dotado de otro botón de RESET, que pone el Spectrum «a cero». Este proceso vuelve a cargar el sistema operativo del disco, siendo equivalente a apagar y encender el ordenador.

Sistema de directorios: Una complicación innecesaria

El sistema operativo, llamado TOS (*Timex Operating System*), atrae a primera vista: El sistema de ficheros tiene estructura de árbol, con los dos puertos RS232 y los directorios de cada disco «creciendo» de la raíz, de los que a su vez salen nuevas ramas (subdirectorios) y hojas (programas). El sistema conoce nuestra posición dentro del árbol en cada instante, y cualquier nombre de fichero se buscará solo en la «rama» en que nos encontremos. Si queremos especificar un fichero que se encuentra en otro subdirectorio, debemos indicar el «camino» que lleva hacia él. Existen también comandos que permiten subir y bajar por el árbol de directorios.

Un sistema operativo con subdirectorios es ideal para el manejo de disco duro donde el espacio disponible hace que se multiplique el número de ficheros, resultando difícil identificar en el catálogo los archivos que nos interesan. El uso de directorios independientes para distintos usuarios o aplicaciones permite una mayor claridad en la gestión del disco. No parece intere-

sante su uso en un ordenador como el Spectrum, con *diskettes* de 140 K. En las pruebas realizadas hemos podido constatar que con este sistema el usuario poco experimentado «pierde» los archivos, ya que no se «atina» con un programa determinado en el «bosque» de directorios.

Una característica más interesante es la posibilidad de usar caracteres como «comodín» en algunos de los comandos, al referirnos a un nombre de archivo. La presencia de un interrogante «?» en el nombre hace que el programa incluya cualquier archivo que tenga en la posición del interrogante un caracter arbitrario: CA? selecciona cualquier fichero de tres caracteres que comience por CA. Si usamos el signo más «+», el programa lo reemplazará por cualquier cadena de caracteres: el nombre «+» es cualquier archivo que no tenga extensión. «+.BAS» selecciona todos los ficheros con extensión BAS. Muy útil para borrar todo aquello que hayamos caracterizado como fichero tipo BASIC en el directorio en vigor.

Los comandos son palabras clave del Spectrum seguidas por un asterisco y sus parámetros, y el sistema de paginación es muy parecido al del *Interface I*, por lo que



ambos sistemas son incompatibles. Existen instrucciones para la carga, almacenamiento y cambio de nombre de archivos, tanto programas, como código máquina o datos. El comando MOVE * permite la copia de archivos o grupos de archivos. El comando ATTR permite la protección de archivos o su ocultación en los directorios. DIM da la posibilidad de crear un archivo o directorio nuevo (vacíos).

Canales y Archivos de datos. Acceso directo

El sistema operativo TOS proporciona 16 canales de acceso a datos, diferentes de los canales del Spectrum. Los cuatro primeros canales tienen un tratamiento distinto, con un *Buffer* de 512 *bytes* para cada uno, y el sistema los distingue como canales «rápidos». Los restantes comparten otros 512 caracteres y, si se abren varios canales

ARCHIVOS DE ACCESO ALEATORIO

La principal ventaja que ofrecen los sistemas de disco sobre otras alternativas de almacenamiento consiste en que el acceso aleatorio es real, y no se simula mediante software. Presentamos un ejemplo: cómo se puede usar el acceso aleatorio para crear un pequeño archivo de direcciones y consultarlo. Para ello, lo primero será crear el archivo, si no existía:

```
DIM * "direc.dat"
```

creará un fichero, dando un error si ya existía.

Vayamos al programa propiamente dicho:

```
100 OPEN  *1; "direc.dat"; r;100
```

abre el archivo al canal 1, con acceso aleatorio. Cada registro tiene 100 caracteres.

```
110 INPUT "1-Consultar, 2-Añadir, 3-Fin"; h$
120 IF h$="2" THEN GOTO 500
130 IF h$="3" THEN GOTO 600
140 IF h$<>"1" THEN GOTO 110
```

Hasta quí hemos comprobado las opciones y hemos enviado el programa al punto adecuado. Veamos a continuación cómo se consulta el archivo:

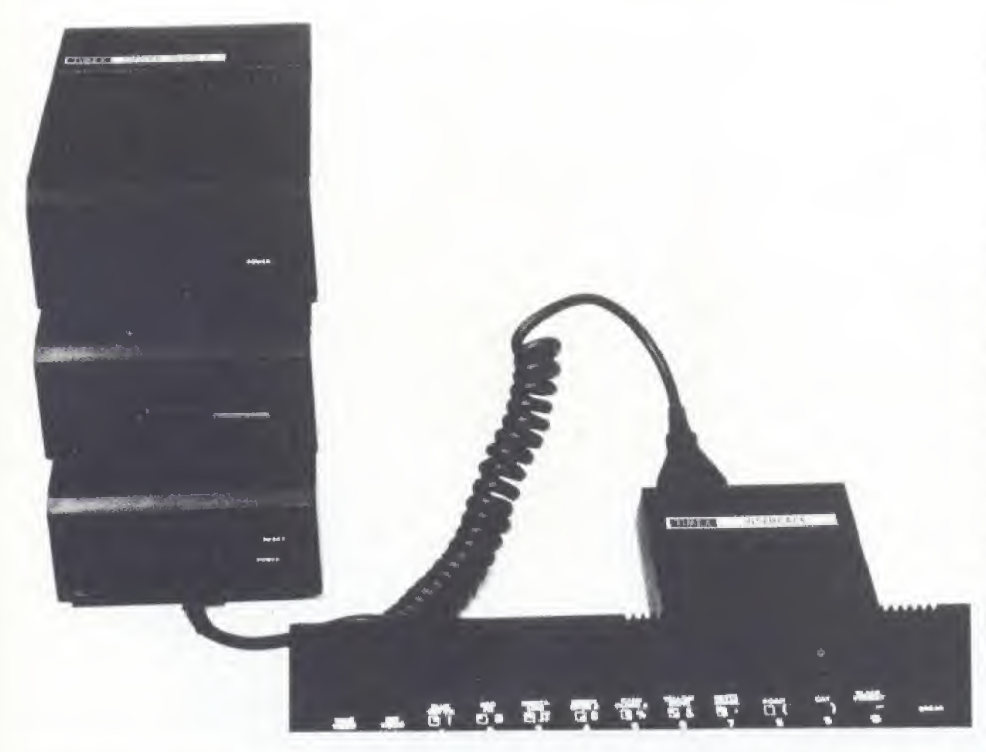
```
150 INPUT "Registro?";n
160 IF n>65535 OR n<1 THEN GOTO 150
```

ya que el número máximo de registros es de 65535, comenzando a numerar por uno.

```
170 INPUT* 1;a$;AT n
```

Esta instrucción lee el registro número n del archivo, y lo introduce en la variable a$.

```
180 CLS
190 PRINT AT 4,3;"Nombre: ";a$ (TO 30)
```



«lentos», los tiempos de acceso serán más altos.

Cada uno de estos nuevos canales se puede abrir a un archivo, de tipo secuencial o de acceso directo, o bien a uno de los *ports* RS232. Para los archivos de acceso directo deberemos especificar la longitud de registro (máximo 256 caracteres). El almacenamiento mediante archivos de acceso aleatorio abre la puerta a aplicaciones rápidas con manejo de gran cantidad de datos, ya que los tiempos de acceso son realmente buenos.

Comunicaciones serie

Dos conexiones RS232 permiten la conexión con otros ordenadores, impresoras serie, o cualquier otro dispositivo de entrada/salida RS232. El sistema operativo permite la configuración de los dos puertos independientemente y con una gran flexibilidad. Los conectores utilizados son tipo *joystick,* no estándar, aunque no aparecen grandes problemas de cableado.

El manual es, junto con el precio, uno de los puntos más débiles de la unidad de **Investrónica.** Un manual en inglés, donde se pueden encontrar todas las características del programa, aunque haya que perder un tiempo excesivo en la búsqueda. Un apéndice explica la implementación de rutinas en código máquina que saquen partido del disco, pero no existe ninguna referencia al uso de vectores para la ampliación del BASIC, presentes en el *Wafadrive* y el *Microdrive.*

Las unidades de 640 K, anunciadas por Timex, parecen haber desaparecido en el negro túnel del marketing. Una lástima, ya que darían mucho más juego a esta unidad. Para finalizar, conviene no olvidar otro problema que surge con las unidades de disco para Spectrum (no sólo la de Investrónica): la escasez de *software* comercial desarrollado para ellas. En general, no queda más remedio que escribir nuestros propios programas o adaptar alguno realizado para microdrive. Y ese es otro aspecto que pesa en la balanza.

```
200 PRINT AT 8,0;"Dirección: ";a$(31 TO 60)
210 PRINT AT 12,5;"Tfno: ";a$(61 TO 75)
220 PRINT AT 16,4;"Notas: "; a$(76 TO )
230 INPUT "Enter para seguir"; LINE h$
```

Estas líneas leen un registro. Por convenio, hemos decidido que los primeros 30 caracteres almacenarán el nombre, después otros 30 caracteres almacenarán la dirección, quince caracteres los dejamos para el número de teléfono y los veinticinco últimos para notas de interés.

```
240 GOTO 110
```

La sección siguiente del programa se encarga de la introducción de las fichas. Primero se nos pregunta en qué registro queremos introducir la ficha. Hay que tener cuidado, ya que el programa no mirará si ese registro contenía algún dato de valor. A continuación el programa dimensiona cuatro cadenas de caracteres, que contendrán los datos de nuestra ficha. Para finalizar, se introducen los datos desde el teclado mediante instrucciones INPUT y se imprime el registro resultante al archivo.

```
500 INPUT "Registro Numero?";n
510 IF n>65535 OR n<1 THEN GOTO 500
520 CLS
530 DIM n$(30): DIM b$(30): DIM p$(15): DIM c$(25)
540 INPUT "Nombre? ";n$
550 INPUT "Dirección? ";b$
560 INPUT "Tfno? ";p$
570 INPUT "Notas? ";c$
580 PRINT * 1;n$+b$+p$+c$;AT n
590 GOTO 110
600 CLOSE  *1: PRINT "Terminado"
```

La línea 600 cierra el archivo, permitiéndonos así conservar nuestras fichas sin problemas.

UNIDADES DE DISCO

Resumimos a continuación los comandos del sistema operativo del Invesdisk. Donde se especifica camino se entiende una referencia cualquiera a un archivo (puede ser necesario indicar en qué directorio está situado, los parámetros entre corchetes son optativos).

ATTR * camino P o U o I o V

La opción P protege el archivo contra escritura. La opción U quita la protección. I hace que el archivo no aparezca en los catálogos. V tiene el efecto contrario. Esta opción admite comodines, por lo que se puede proteger un directorio completo, o bien ocultar parte de los ficheros.

CAT * [camino]

Imprime información sobre el directorio activo o bien sobre el directorio indicado mediante camino.

CLOSE # * expresión

Cierra el canal indicado por la expresión numérica.

DIM * camino

Permite crear un archivo o directorio (si lleva la extensión .DIR) especificado mediante su nombre.

DRAW *

Esta instrucción equivale al RETURN después de un GO SUB *. Sirve para activar el directorio previo.

ERASE * camino [N]

Borra el archivo especificado. Mediante comodines se pueden borrar varios archivos en una sola operación. La opción n hace que el programa no nos pida confirmación en cada archivo.

FORMAT * unidad TO nombre

Inicializa el disco presente en la unidad especificada (A, B, C o D). El nombre indicado pasa a ser el directorio principal de ese disco. La operación destruye todos los ficheros presentes.

FORMAT * puerto-serie

Esta opción se utiliza para redefinir los parámetros asociados con el protocolo de transmisión RS 232. Los cambios se pueden hacer permanentes salvando en disco la nueva versión del operativo.

GO SUB * [camino] o GO SUB * nombre d

En su primera forma permite convertir en activo el directorio deseado, «recordando» el directorio anterior, por lo que se podrá volver. En la segunda forma selecciona el directorio principal de una de las cuatro unidades.

GO TO * camino o GO TO * nombre d

Igual de GO SUB, pero no conserva registro del directorio activo.

INPUT * # n; var $ [;AT p]

Lee un caracter o un registro del archivo conectado al canal n, y lo deja en la variable var$. Si el archivo es de acceso aleatorio p indica qué registro debe ser leído.

LET * camino1 TO camino2

Permite cambiar de nombre a archivos, directorios o puertos serie.

LIST *

Imprime información sobre el directorio activo y los anteriores (si se ha usado el comando GO SUB *).

LIST * # [N]

Informa sobre el número de canales abiertos (sin parámetro) o proporciona información sobre el canal n.

LOAD * camino [opciones]

Similar a la instrucción de *cassette.* La única diferencia es que la especificación de nombre puede incluir subdirectorios.

MERGE * camino

Similar al Spectrum.

MOVE * camino-origen TO camino- destino

Copia un archivo o puerto serie a otro archivo o puerto serie. Se pueden utilizar comodines, aunque en ese caso el nombre destino debe ser un directorio.

OPEN # * expr1; camino; modo [expr2]

Abre el canal indicado por expr1 al archivo indicado por camino. El modo puede ser de entrada, de salida, de acceso aleatorio o adicionar texto a un archivo existente. Si el acceso es aleatorio, expr2 indica la longitud de registro.

PRINT * # n; str$; [;AT n]

Imprime al canal indicado la cadena de caracteres str$. Si el archivo es de acceso directo se puede indicar en qué registro se quiere escribir.

RESTORE* n

Sitúa el puntero del archivo asociado al canal n al comienzo del archivo. En modo entrada permite volver a leer un archivo. En modo salida lo borra y comienza a escribir encima. En modo adición permite reescribir el comienzo de un archivo manteniendo el resto. No es necesario en acceso aleatorio.

SAVE * camino opciones [N]

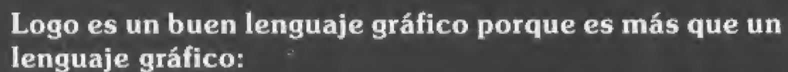
Análoga a la instrucción del Spectrum. Si el archivo ya existía el programa nos da la opción de borrarlo excepto cuando incluimos el parámetro N.



¡¡YA ESTA AQUI EL LOGO SINCLAIR EN CASTELLANO PARA TU SPECTRUM 48 K Y PLUS!!

RT. - AV. . .
N, PONPOS . . .
GIRA DERECHA,
ELIZA

Logo es un buen lenguaje para los niños porque es un buen lenguaje para todo el mundo: (niños desde los 4 a los 90 años)
- un lenguaje sin límites: accesible al joven principiante, y al mismo tiempo potente como para estimular a un programador experimentado.
- un lenguaje simple de abordar: te comunicas con el ordenador en castellano... (por fin un lenguaje de programación en castellano).



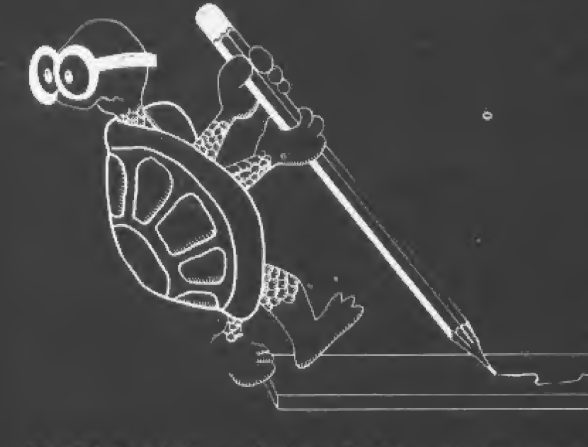
Logo es un buen lenguaje gráfico porque es más que un lenguaje gráfico:
- un lenguaje que ofrece la posibilidad de manipular listas, palabras, operaciones aritméticas, contribuye a un grafismo elaborado.
- un lenguaje que permite iniciarse en la programación a través del grafismo en el que los resultados son concretos y visibles.

Logo es un buen lenguaje porque es un lenguaje potente:
- un lenguaje interactivo: las instrucciones son ejecutadas inmediatamente y los mensajes de ayuda son claros y precisos.
- un lenguaje que permite describir las acciones a ejecutar de forma estructurada, creando nuevos procedimientos a partir de las instrucciones iniciales (primitivas).

Usar Logo es aprender:
- numerosas experiencias pedagógicas lo han demostrado: Logo es una herramienta de expresión que incita a reflexionar sobre la propia metodología.
- un lenguaje que permite que cada cual domine el ordenador en función de sus necesidades: programas de aplicación, educativos, de juegos...

Y ADEMAS...

LE DAMOS AYUDA AL LOGOADICTO CON: **EL LOGOSPECTRUM CLUB,** que te permitirá realizar todo tipo de consultas. Con sólo enviar el cupón que acompaña al programa, recibirás a vuelta de correo tu clave de usuario. Este servicio es gratuito.



EDICION LIMITADA DE LANZAMIENTO

Hemos preparado una edición limitada y a la mitad de precio que este programa tiene en Inglaterra.

¡¡HAZTE AMIGO DEL LOGO!!
por sólo **3.990 pts.**

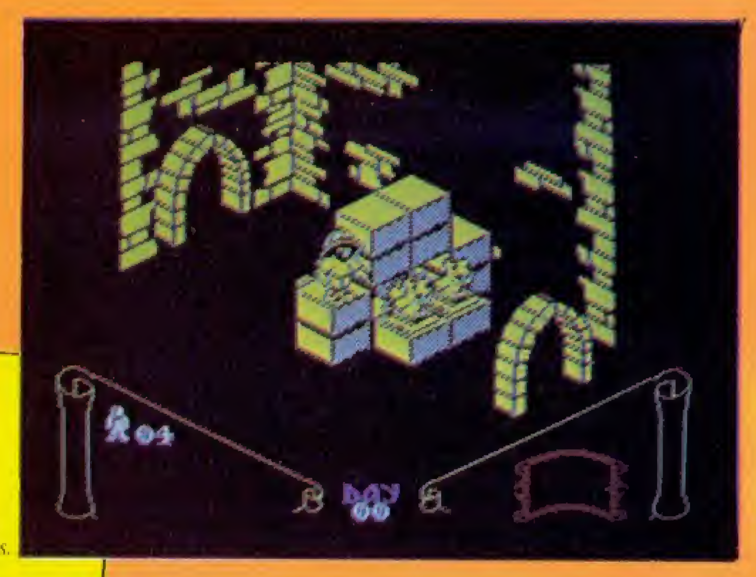
FABRICACION, ASISTENCIA TECNICA Y CONSULTAS DEL LOGO SPECTRUM CLUB, DIRIGIRSE A:
IOSHUA
Provenza, 281, 2.º, 5 - Tel. 215 83 37
08037 BARCELONA

COMERCIALIZACION
VENTAMATIC
Córcega, 89, ent. - Tel. 230 97 90
08029 BARCELONA

Juegos

KNIGHT LORE

Erbe
Spectrum 48 K
Precio: 2.000 Ptas.



Sobreman ha de recorrer el laberinto. En ocasiones desplazando bloques para saltar sobre ellos.



Es medianoche y la transformación ha tenido lugar: el hombre-lobo busca la solución a su «tormento».

Hay casas de *software* que van haciendo historia por la calidad de sus juegos y la consecuente expectación que genera la aparición de los nuevos productos. Tal es el caso de la casa inglesa Ultimate, donde el «premio» por llegar al final de sus juegos es precisamente el poder conocer el nombre del próximo juego que lanzarán al mercado.

Knight Lore, como todo lo de Ultimate, sigue la línea del laberinto, de la sucesión de cuidadas pantallas en búsqueda de los distintos materiales que ayudan a llegar al final. En este caso Sabreman —protagonista usual de los juegos de Ultimate— se encuentra con el problema de convertirse en hombre-lobo en los días de luna llena. Para resolver su problema ha de recoger diversos ingredientes —esparcidos por un tétrico castillo— y llegar con ellos a donde se encuentra un mago que conoce la combinación que le liberará de su castigo.

Aunque pueda parecer sencillo, el problema es en realidad una lucha contra reloj en dos sentidos: sólo se dispone de 40 días y 40 noches para poder «deshacerse» del problema; y cada vez que tiene lugar la transformación (de una vistosidad realmente inmejorable), nuestro héroe queda inhabilitado para defenderse de cualquier ataque, pudiendo perder alguna de las cinco vidas que dispone al iniciar el juego.

Los objetos transportados aparecen en la parte inferior de la pantalla pudiéndose coger y soltar cualquiera a voluntad y llevando un máximo de tres, lo que nos recuerda al ya clásico Saimazon. Asimismo, la parte inferior de la pantalla informa también del número de días transcurridos y la trayectoria del sol o la luna, imprescindible para estar preparado ante la «transformación».

La clave del juego consiste en conocer la secuencia en que se han de arrojar los catorce objetos a la pócima del mago, a fin de librarse del hechizo. Aunque pueda parecer aleatorio, afortunadamente no es este el caso y con la adicción que genera el juego se puede llegar a conocer.

Control: Teclado, joystick.

Jugadores: Uno.

Gráficos: Lo mejor, como viene siendo habitual en Ultimate.

Nivel de dificultad: No existen.

Originalidad: Máxima originalidad y perfecta transformación en hombre - lobo - hombre.

Conclusión: Adictivo para todos aquellos que disfruten con este tipo de juegos-laberinto, en los que pasar a otra habitación es pasar a una situación completamente diferente.

SKOOL DAZE

Serma
Spectrum 48 K
2.500 ptas.

«¡Atrévete a todo!», «¡Juega con tus profesores!», son, entre otros, los titulares agresivos con que la publicidad nos enseña lo más novedoso en juegos para Spectrum.

Y ciertamente se trata de un juego donde su mayor atractivo reside en su originalidad. A diferencia de Knight Lore y la serie de juegos basados en múltiples pantallas, aquí se limita todo a tres pantallas. Lo importante no es andar por recónditos caminos, sino ir al colegio con el aliciente de hacer todo lo que se le ocurra a su imaginación: pegarse con los compañeros o los profesores, abandonar las clases, meterse en el despacho del director y, por supuesto, abrir la caja fuerte donde están las notas... ¡Ya se puede imaginar con qué motivo! Este es precisamente el objetivo del juego, pero no es tan fácil como traspasar la puerta. Antes, hay que golpear con el tirachinas a los escudos del colegio, lo cual sólo es posible de «rebote» una vez golpeado a los profesores.

Toda la aventura transcurre en lo que pudièramos calificar como «un día normal de escuela», donde Eric, que es el protagonista del juego, ha de asistir a varias clases.

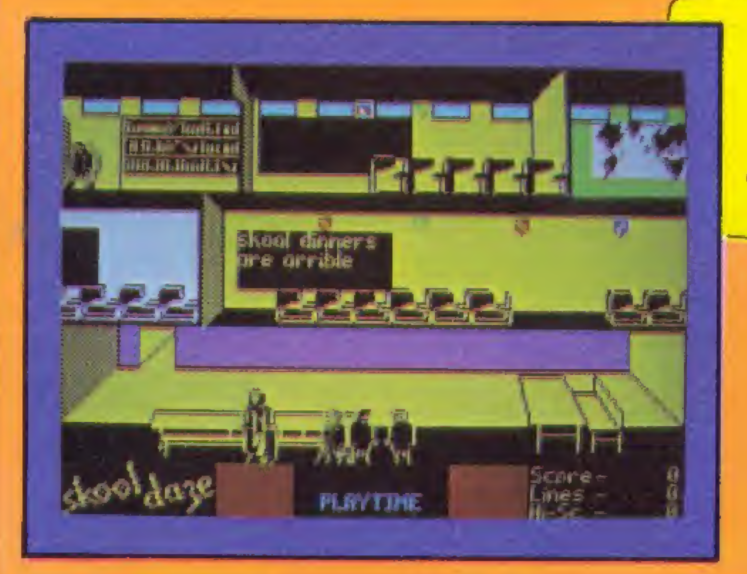
La pantalla del televisor se convierte en un colegio lleno de bullicio y movimiento por doquier (cuando se ve por primera vez siempre surge la pregunta: ¿cómo han logrado tener todo esto en movimiento?)

Al margen del movimiento, existe una característica que le hace especialmente novedoso, cual es la conversación profesor-alumnos al estilo de los comics, incluido el chivato que no duda en advertir al profesor de la ausencia de Eric.

La limitación de tiempo viene impuesta por los castigos de los «profes» empeñados en hacer copiar numerosas líneas sobre lo que no se debe hacer. Cuando el número de líneas rebasa la cifra de 100, el director decide «dar vacaciones» a Eric y no le permite volver al colegio hasta que las haya copiado todas. Aunque también puede ser expulsado prematuramente por coger las paperas.

Un juego original y recomendable «para todos los públicos» y poderse finalmente ensañar con los profesores y con los diabólicos compañeros de escuela.

Control: Teclado, joystick.
Jugadores: Uno.
Gráficos: Muy buenos, tanto por la caracterización de los distintos personajes, como por el movimiento de todos ellos en pantalla. El desplazamiento horizontal pixel a pixel está bien logrado.
Niveles de dificultad: No existen.
Originalidad: Este juego rompe con los clásicos de «innumerables» pantallas. Pocas pantallas, mucha acción y muchos golpes para los profesores.
Conclusión: A pesar de los problemas del idioma inglés, es un juego tremendamente adictivo. Lógicamente la técnica de mensajes en pantalla simulando los comics, no resulta de fácil traducción, afortunadamente completado con buenas instrucciones un buen manual de instrucciones.



Es «playtime», es decir, recreo. En la pizarra se puede leer «las cenas de la escuela son horribles».



Eric ha sido cogido fuera de la clase en que debía estar. El castigo es escribir 600 líneas.

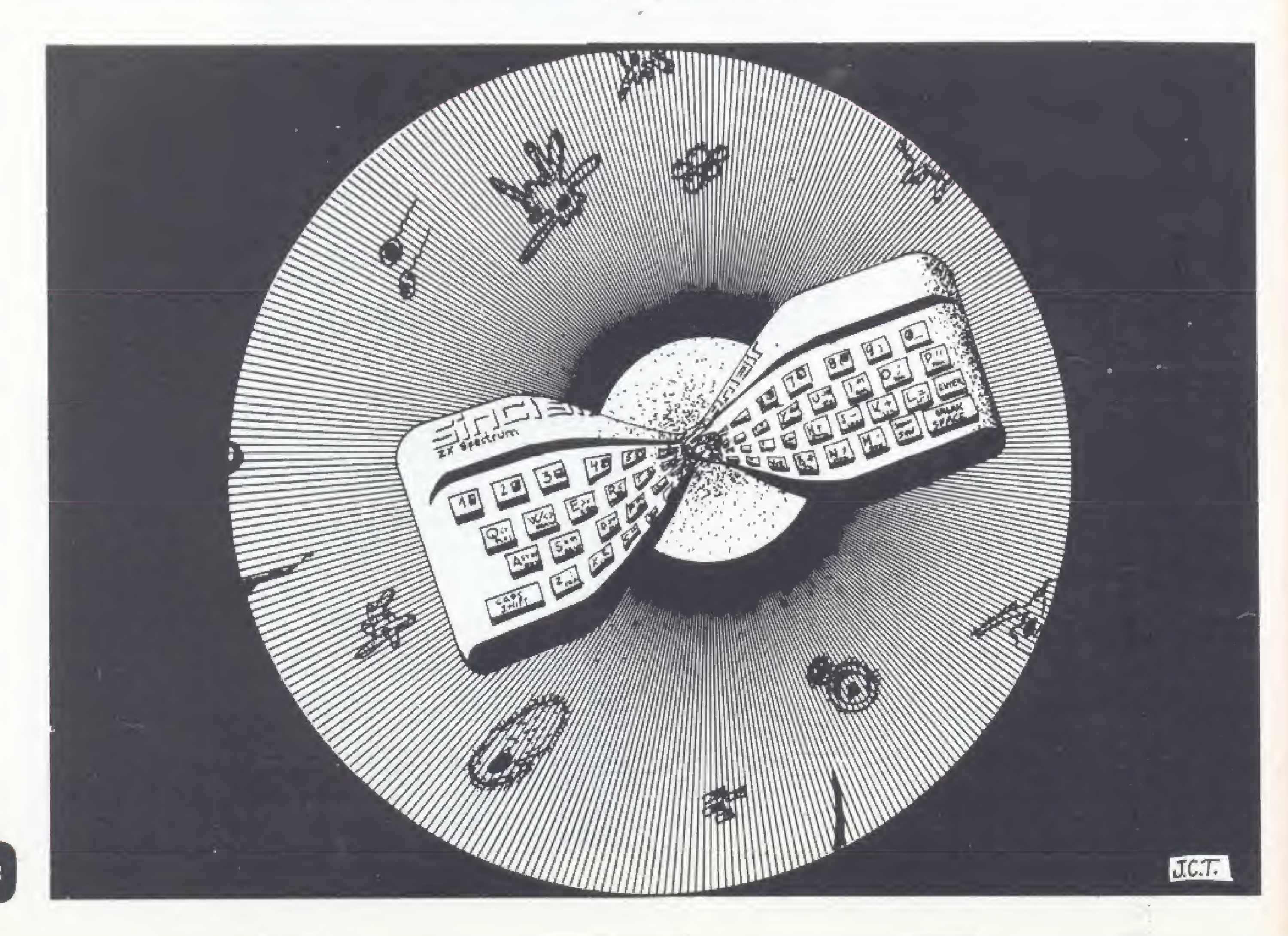
2 PROGRA SI

Esta corta subrutina en código máquina nos permitirá acceder a dos programas BASIC simultáneamente, siempre que cada uno de ellos no supere las 20 K.

El método es sencillo: se crea, si no se había hecho antes o si hemos hecho un NEW (VARS2=0), un segundo programa vacío en la parte alta de la memoria (EFRIA), y sus respectivos PROG, VARS y ELINE se almacenan como PROG2, VARS2 y ELIN2. Cada vez que cambiemos de programa se intercambiarán entre sí estas variables (CAMBIO). Al final (SALIDA) se hace un listado automático (CALL AUTO-LIST) y tras poner a cero el registro A se retorna (JP MAING-G) con mensaje O OK. Entrando por 65008 (BORRAR) borraremos el programa en que estemos con sus variables, poniendo a sus mínimos valores VARS y ELINE.

Quien no tenga ensamblador deberá teclear el programa cargador, que avisará, si es incorrecta la suma de control, con el mensaje «ERROR EN DATA»; en cuyo caso deberemos revisar los datos y ejecutarlo de nuevo.

Antes de utilizar la subrutina



MAS BASIC
MULTANEOS

conviene grabar el programa en cinta o microdrive. para salvarlo en forma de *bytes* haremos: SAVE «nombre»CODE 65000. 80 y para cargarlo posteriormente: CLEAR 64999: LOAD «nombre»CODE.

Seguidamente podremos borrar el programa cargador con RANDOMIZE USR 65008 o con NEW e introducir el primer programa BASIC de la forma habitual. Cuando hayamos acabado deberemos hacer: RANDOMIZE USR 65000 y estaremos en condiciones de introducir el segundo programa, repitiéndolo volveremos al primero, y así sucesivamente. Si queremos borrar uno de los dos programas manteniendo el otro, podemos usar RANDOMIZE USR 65008. Si hacemos NEW se borrarán los dos programas, pero no así el código máquina, que podrá ser utilizado normalmente.

Cada programa conserva sus propias variables, pero no la pila de GO SUB, que se borra en cada cambio.

Quien tenga el interface 1 debe tener cuidado de no hacer operaciones con el microdrive desde el segundo programa, pues desplazaría al primero y al volver a éste, podría "colgar" el sistema.

Luis Gala

PROGRAMA 1

```
  10 REM CARGADOR
  20 REM
  30 REM DOS PROG. SIMULTANEOS
  40 REM
  50 CLEAR 64999
  60 LET C=0
  70 FOR N=0 TO 79
  80 READ A
  90 LET C=C+A
 100 POKE 65000+N,A
 110 NEXT N
 120 IF C<>9447 THEN PRINT FLASH
 1;" ERROR EN DATA ": STOP
 130 PRINT "CARGA OK"'" RANDOMIZ
E USR 65000 PARA CAMBIO DE PROGR
AMA. 65008 PARA BORRAR."
 140 DATA 205,9,254,205,149,23
 150 DATA 175,195,19,19,205,9
 160 DATA 254,205,0,254,24,241
 170 DATA 33,200,175,34,56,254
 180 DATA 34,176,92,54,128,35
 190 DATA 34,58,254,237,75,176
 200 DATA 92,4,5,40,233,42,75
 210 DATA 92,34,176,92,237,67
 220 DATA 75,92,237,75,58,254
 230 DATA 42,89,92,34,58,254
 240 DATA 237,67,89,92,237,75
 250 DATA 56,254,42,83,92,34,56
 260 DATA 254,237,67,83,92,201
```

PROGRAMA 2

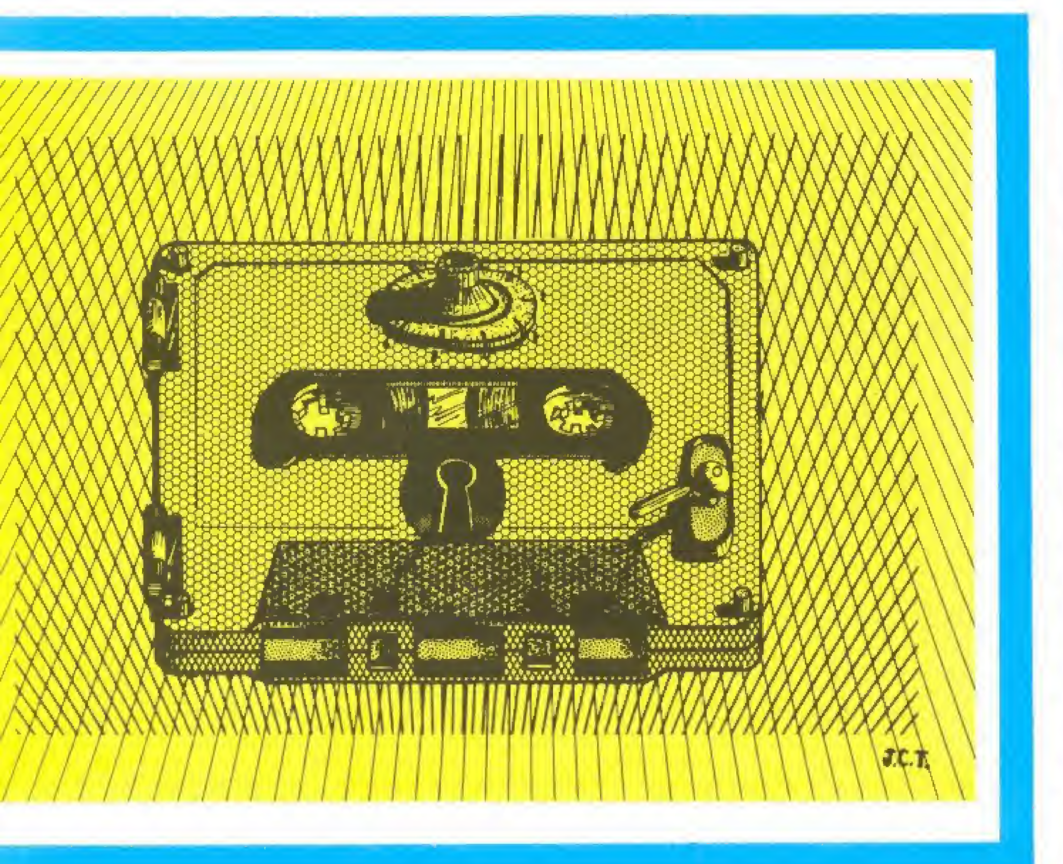
```
 10          ORG  #FDE8
 20 ENTR     CALL CAMBIO
 30 SALIDA   CALL #1795
 40          XOR  A
 50          JP   #1313
 60 BORRAR   CALL CAMBIO
 70          CALL BORR
 80          JR   SALIDA
 90 EFRIA    LD   HL,#AFC8
100          LD   (PROG2),HL
110 BORR     LD   (VARS2),HL
120          LD   (HL),#80
130          INC  HL
140          LD   (ELIN2),HL
150 CAMBIO   LD   BC,(VARS2)
160          INC  B
170          DEC  B
180          JR   Z,EFRIA
190          LD   HL,(VARS)
200          LD   (VARS2),HL
210          LD   (VARS),BC
220          LD   BC,(ELIN2)
230          LD   HL,(ELINE)
240          LD   (ELIN2),HL
250          LD   (ELINE),BC
260          LD   BC,(PROG2)
270          LD   HL,(PROG)
280          LD   (PROG2),HL
290          LD   (PROG),BC
300          RET
310 PROG2    DEFS 2
320 ELIN2    DEFS 2
330 VARS2    EQU  #5CB0
340 PROG     EQU  #5C53
350 ELINE    EQU  #5C59
360 VARS     EQU  #5C4B
```



PROTECCION

A medida que aumenta el número de usuarios del ya enomemente popular Spectrum, la piratería se convierte cada vez más en un problema acuciante para las jóvenes casas de *software.* Los usuarios alcanzan un grado de conocimiento de sus máquinas tal que no supone problema alguno para ellos «destripar» el más protegido de los programas. Los programadores profesionales se ven obligados a forzar su imaginación al máximo para desarrollar sistemas que impidan la copia de su preciado *software.* Y aunque, como todos sabemos, no exista programa soportado en cinta de *cassete* que no sea vulnerable a la copia, las técnicas empleadas comienzan a extenderse incluso al campo de la protección contra algunos de estos sistemas. Al menos se han desarrollado algunas ideas que impiden (nunca con una fiabilidad absoluta) la copia de cintas con dos aparatos cuyo nivel de grabación esté regulado con ALC (*Automatic Level Control* o control automático de nivel), sistema usado por la práctica totalidad de los *cassetes* portátiles. Y es ésta una de las formas mediante las cuales los usuarios realizan la mayor parte de las copias piratas. Contra los que se dedican a este sucio negocio de una forma profesional, nada se puede hacer. La única forma de reducir esta piratería es simplemente abandonando la cinta como soporte de *software.*

De las últimas técnicas desarrolladas por las casas de *software* es de lo que se va a hablar aquí, y para ello nos basaremos en uno de los juegos más populares de los editados recientemente para el Spectrum: el DECATHLON, de la compañía inglesa OCEAN. Previamente daremos un breve repaso a los sistemas de protección más usuales, de los cuales se ha escrito bastante (ver TODOSPECTRUM número 1).



Entre los sistemas de protección empleados podemos separar grandes grupos según la finalidad de la técnica usada. Por una parte se suelen proteger los programas contra la posibilidad de transformar o usar rutinas contenidas en el *software.* Por otra, la protección es contra la copia mediante los conocidos programas copiadores, de los que existe ya una gran variedad en el mercado. El primer conjunto realiza también en parte la función de evitar la copia directa por análisis de las instrucciones o rutinas de carga que posee el programa. Esto se consigue en primer lugar haciendo imparable el programa. Todos sabemos lo que suele ocurrir si intentamos hacer un «Break» durante la carga o ejecución de la mayoría de los programas comerciales. Para ello, se ha de alterar o anular la función de la tecla «Break». Son varios los métodos empleados, entre los que cabe destacar la alteración de las variables del sistema «ERROR SP» o «DEF SZ». La primera es un puntero de dirección que indica la posición de memoria donde se halla la dirección de la rutina que ha de emplear el sistema operativo en caso de que se produzca un error. Si cambiamos este valor podemos hacer que el ordenador se bloquee al producirse una interrupción, ya que intentará ejecutar una rutina que no es tal. La segunda variable

DE SOFTWARE

indica al sistema la cantidad de líneas que puede usar en la parte baja de la pantalla para mensajes y entrada de datos. Si se hace cero, cuando el ordenador intente dar el mensaje de «BREAK IN LINE xx» no podrá hacerlo y se bloqueará. Sin embargo, no es un secreto que es posible parar la autoejecución de un programa cargándolo con la función MERGE en lugar de con LOAD. MERGE fusiona dos programas en BASIC alterando si es necesario la numeración de las líneas, motivo por el cual, anula la autoejecución. El remedio contra este truco consiste en alterar los números de línea sustituyéndolos por otros «imposibles» para el ordenador, como el cero, etc. De esta forma se consigue además que la línea no se pueda editar por medios normales. Cuando el ordenador intente renumerar el programa, se encontrará con números incomprensibles y se bloqueará. Todo esto, por desgracia, tiene un antídoto bastante sencillo para el programador de código máquina. La autoejecución de un programa viene definida por dos bytes de la cabecera que indican la línea de comienzo del programa. Si este número es mayor de 32768 no ocurre autoejecución. Basta, por tanto, con crear una cabecera igual a la del programa pero sin autoejecución, llamando directamente a las rutinas de carga y grabación de bloques de la ROM. Dado que es posible, como se ve, parar la ejecución del programa a pesar de todo, es necesario proteger el propio programa contra su alteración o visualización en pantalla. Una forma bastante sencilla es usando los códigos de color dentro de las líneas de programa, tal y como se explica en el manual del Spectrum. Se pueden poner tinta y papel del mismo color, de forma que no aparezca nada al listar el programa. Para evitar la edición y supresión de estos códigos en la línea se emplea de nuevo el truco de la línea cero. «Pokeando» ceros en los *bytes* correspondientes al número de línea en cada sentencia, se conseguirá hacer las líneas ineditables. Basta con deshacer el entuerto para obtener un programa normal. En última instancia se intenta proteger la parte en código máquina que llevan la mayoría de los programas. Para ello los bloques correspondientes se suelen cargar mediante rutinas en máquina extrañas para los profanos, y a las cuales se accede no con un vulgar RAND USR, sino mediante alteración de las variables del sistema (ERROR SP). De esta forma es difícil conocer dónde comienza la dirección de ejecución del código, y por tanto se evita su desensamblado.

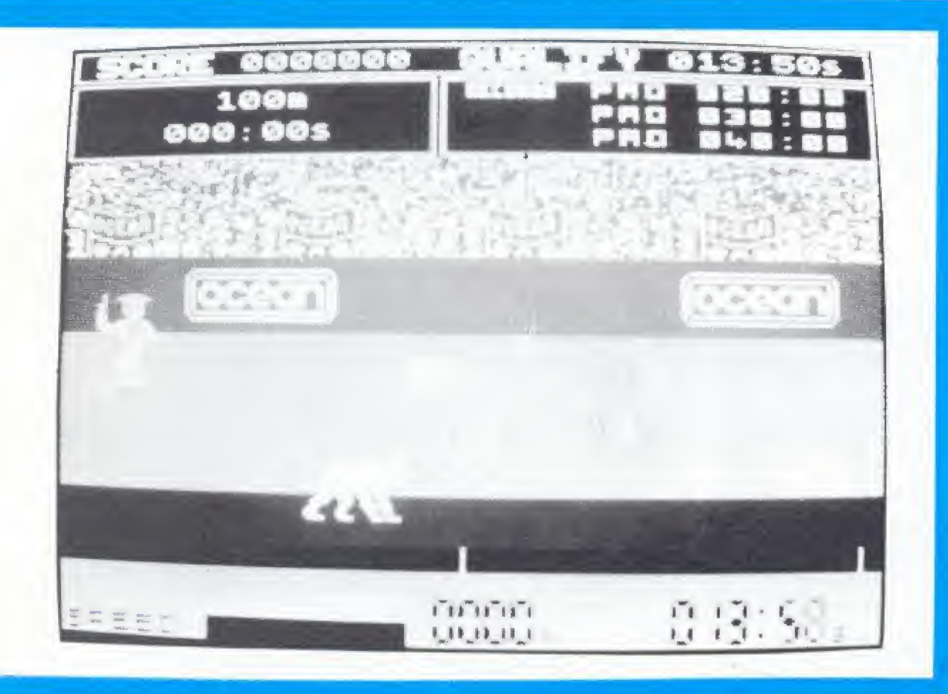
Las casas de software van por delante de las técnicas utilizadas en los programas copiadores. Cuando apareció el famoso «the key», muchos programas estaban ya grabados con bloques sin cabecera.

Respecto a la protección contra los conocidos copiadores, hay que reconocer que muchas casas de *software* han ido siempre, por delante de éstos en técnicas de protección. Cuando el único copiador conocido en el mercado era el famoso THE KEY, muchos programas estaban ya grabados con bloques sin cabecera, imposibles de leer por el copiador. Aparecieron programas que copiaban estos bloques, pero para entonces el *software* venía protegido con falsas cabeceras que engañaban astutamente al copiador. Recientemente han aparecido en el mercado varios programas que parece ser han conseguido desterrar del «reino» del Spectrum a los más potentes copiadores. Son los llamados programas «turbo», grabados a distinta velocidad del estándar prefijado por las rutinas de la ROM. El cambio de velocidad se consigue alterando ciertas constantes contenidas en las rutinas de carga y grabación. Aparentemente se puede ha-



cer un copiador de velocidad variable de esta forma, pero es prácticamente imposible acertar la velocidad a la que está grabado un programa. Técnicamente es posible escribir una rutina que la mida, pero es bastante complejo. Probablemente ya haya gente trabajando en ello.

De las últimas técnicas de protección es un claro y brillante exponente el conocido DECATHLON de OCEAN. Los programadores de esta casa han conseguido un sistema realmente efectivo, aunque como veremos no es totalmente invulnerable. Los sistemas empleados son novedosos tanto en el campo de la protección del BASIC y del código máquina como en el de la copia con programas copiadores y directamente de cinta a cinta.



Analizando en profundidad este programa, nos encontramos con que consta únicamente de 3 bloques. El primero es un bloque BASIC, seguido inmediatamente de dos bloques sin cabecera grabados ya a distinta velocidad. Este hecho no llamaría la atención si no fuese porque al parar el programa y observar el BASIC nos encontramos con que no existe ninguna línea REM que contenga rutinas en máquina. Evidentemente, éstas han de existir para cargar bloques a distinta velocidad. Pero no están metidas como es corriente en una línea de programa, sino que se encuentran ubicadas en la zona de variables, único lugar remanente protegido de las inclemencias del sistema operativo. El programa en BASIC tiene todas sus líneas numeradas con cero y con códigos de color que las hacen invisibles en los listados. El MERGE, por tanto, es inútil, siendo necesario alterar la cabecera para cargar sin autoejecución. Esto ha de hacerse llamando a la rutina de carga de bloques para cargar la cabecera, alterando los dos bytes de ejecución, y grabándola de nuevo con otra llamada a la ROM. Para conseguir visualizar el programa es necesario anular los códigos de color mediante un monitor en máquina ubicado en una zona alta de la memoria, ya que es difícil hacerlo desde BASIC sín trastocar la zona de variables, y por tanto el código que contiene. Los números de línea se pueden cambiar de igual modo. Con el programa al descubierto, nos encontramos con una maraña de PEEK y POKE que hacen ciertamente difícil la comprensión de su funcionamiento. En la primera línea, se altera la variable del sistema DEF SZ, que hace el programa imparable. Es difícil cambiar el programa sin anular su correcto funcionamiento, pues si se reduce o aumenta su tamaño se altera la direccion de las rutinas en máquina. Si se sigue paso a paso el proceso de PEEK y POKE de cada sentencia, probablemente se llegue a la conclusión de que lo único que hace el programa es alterar la información de canales, cuando en realidad está cambiando la variable ERROR SP. Esto demuestra la astucia del programador. Con la variable de puntero de error alterada, el sistema salta, una vez terminado el BASIC, a una rutina de la zona de variables con la sencilla función de desplazar un bloque de memoria, pero de una forma un tanto especial. Y es aquí donde empieza lo realmente curioso de este programa. Probablemente muchos os quedaréis bastante asombrados si se os dice que existen instrucciones del microprocesador Z-80 que no forman parte del juego estándar y que no figuran en ningún texto sobre él. Pues bien, existen, y precisamente este programa hace uso de ellas para conseguir un código impenetrable. Estas instrucciones se refieren a manipulaciones con el registro índice IY. Como muchos sabéis, no es posible hacer cargas u operaciones directa e independientemente sobre los registros I o Y (con I no me refiero al registro vector de interrupción, sino al *byte* alto del IY). Sin embargo, si hacemos preceder los códigos de operaciones de los registros H o L con el *byte* FD hexadecimal, conseguimos efectuar la misma instrucción sobre el registro I o el Y respectivamente. La rutina en cuestión desplaza el programa cargador en alta velocidad desde la zona de variables a la parte superior de la memoria, saltando posteriormente a ejecutar éste. Todo ello está enmascarado con estas instrucciones no estandar, de forma que cual-

quier desensamblador dará un código erróneo o sin sentido. Por si fuera poco, la rutina hace uso de *bytes* almacenados en el *stack* antes de su ejecución para calcular la dirección de origen del desplazamiento. Así mismo, y utilizando el registro IY como ya se ha indicado, introduce en el *stack* la dirección de comienzo de la rutina de carga, para saltar a ella posteriormente con un RET, haciendo indescifrable el destino del salto. Todo esto sería inútil si no fuese porque el programa cargador de alta velocidad, que se halla en un principio en la zona de variables, está convenientemente codificado para evitar su desensamblado. Utilizando una técnica similar a la de conversión de código binario en código Gray, la rutina de carga está transformada mediante un OR lógico exclusivo (XOR) entre cada dos *bytes* adyacentes. De igual forma es decodificada por el programa que la desplaza.

El programa cargador es prácticamente una copia de la rutina existente en la ROM con las constantes de tiempo modificadas. Pero también posee algunas particularidades que le hacen realmente ingenioso. En primer lugar carga un bloque de unos 20 *bytes* que no forma parte realmente del programa. Posteriormente carga el bloque principal, que comienza en la dirección 16384, y que por tanto incluye la pantalla. Entre ellos existe una pequeña pausa similar a la de cualquier programa con cabecera, y que no parece significativa en absoluto. Pero en ella reside el secreto de la protección contra la copia de cinta a cinta, por cierto bastante efectiva, como muchos habrán podido comprobar amargamente. Al terminar de cargar el primer bloque, la rutina entra en un bucle de lectura de la puerta del *cassete* con una duración determinada. Para explicar la finalidad de esta lectura, hay que entender el funcionamiento de un *cassete* con control automático de nivel. Este sistema consiste en un amplificador de ganancia variable controlada por tensión con un sistema de ajuste. Si la señal en la entrada es insuficiente para ser grabada, el sistema eleva la ganancia del amplificador hasta que la señal tenga nivel suficiente para la cabeza de grabación. Por el contrario, si la señal es muy fuerte, el amplificador reduce su ganancia hasta el nivel adecuado. En ausencia de señal, el amplificador eleva su ganancia hasta el máximo haciéndose muy sensible. Todos los ruidos que se generan en los circuitos y en las líneas de transmisión son amplificados y grabados con un nivel considerable. Este es el ruido típico que aparece en una grabación con estos aparatos cuando no introducimos ninguna señal. Si hacemos una copia del programa en cuestión con estos *cassetes*, al llegar a la pausa entre los dos bloques el amplificador de grabación elevará su ganancia y grabará los ruidos existentes. El bucle de lectura de la puerta del *cassete* comprueba que no haya ningún ruido extraño. Si lo encuentra, dado que el original debe estar exento de ellos, fija un *byte* de control que hará que el programa se autodestruya al final de la carga, para mayor despiste del «personal».

Este programa, como hemos comprobado, se merece realmente un elogio por la creatividad y el ingenio de sus sistemas de protección. Sin embargo, también hemos visto su vulnerabilidad ante un usuario conocedor de la máquina. Verdaderamente el *software* soportado en *cassete* está condenado definitivamente a la piratería, y quizá la única salida sea el soporte «duro» (ROM) o los sistemas de protección mediante *hardware*, dado que el *microdrive* también es vulnerable con un conocimiento profundo de su sistema operativo. En protección por *hardware* se comienzan a realizar algunos tímidos intentos, pero realmente no es aplicable en programas como juegos, puesto que verían incrementado su precio enormemente.

Javier Cancela

El Decathlon incorpora las mejores técnicas de protección. Un bloque de BASIC seguido de dos bloques sin cabecera.



Conozca Extremadura: CONSULTE A SU

Con motivo de la VII Edición de la Semana de Extramadura en la escuela, se propuso la creación de un programa de ordenador que contribuyera a dar a conocer algunos aspectos de nuestra geografía. Para ello, se eligió un ordenador de amplia difusión Spectrum 48K. Después, se revisó el software *útil para la elaboración del programa Country 48K, Melbourne Draw y Video Display. El primero, sirvió de base para la realización del programa y los dos restantes, se utilizaron para los gráficos.*
Finalmente, se iniciaron los trabajos de recopilación de datos, contando con la Delegación Provincial del Instituto Nacional de Estadística, tanto de Cáceres como de Badajoz.
El resultado final fue este programa.

El programa contiene información sobre todos los municipios extremeños (situación, número de habitantes, distancia a la capital, etc.). Tiene una pantalla principal a todo color y con efectos musicales, que incita a su utilización; y puede ser utilizado para obtener información o para efectuar un examen al usuario.

La información (que se presenta gráficamente), puede ser suministrada de manera automática (aparecen todos los pueblos sucesivamente por orden alfabético) o bien de forma selectiva (para ello el usuario elige el pueblo del que so-



Examen de Geografía, para aprender la localización geográfica de los distintos pueblos.



Información sobre otra población, en este caso de Albala del Caudillo.


Tres niveles de dificultad: EGB, Bachillerato y Especialista.



Información sobre los pueblos: población, extensión, altitud, km. a la capital e índice de crecimiento.

ORDENADOR

licita información); para evitar errores de transcripción, es el propio ordenador el que suministra la lista de los 380 municipios extremeños y el usuario sólo ha de elegir el número correspondiente al pueblo seleccionado. Como esta lista no cabe en la pantalla (sólo se visualizan 10 nombres, simultáneamente con el menú), existen opciones que permiten subir o bajar en la lista total hasta localizar el número correspondiente al pueblo seleccionado.

Cuando se elige la opción de examen, el micro, pide el nivel de dificultad (EGB, Bachillerato o Especialistas) y el nombre del examinado. Mediante un número aleatorio selecciona un pueblo y dibuja una flecha que lo señala en el mapa. También mediante números aleatorios, escribe, en un lateral de la pantalla, cinco nombres de pueblos, convenientemente numerados, entre los que se encuentra el del pueblo que existe en el lugar señalado en el mapa. Dependiendo del índice de dificultad elegido se desprecian aquellos pueblos demasiado pequeños o demasiado cerca del correspondiente a la respuesta correcta. El usuario debe pulsar el número (del 1 al 5) que corresponde al pueblo que, según él, está siendo señalado.

El programa va memorizando el número de respuestas correctas seguidas y presentando el valor del récord y el nombre de la persona que lo ha conseguido. Cuando una respuesta es errónea, da la respuesta correcta y pone a cero el marcador de calificación.

La elaboración de este programa ha pasado por diversas fases y perfeccionamientos, se han ido ampliando las opciones del menú, se han ido incluyendo efectos sonoros y musicales, etc., hasta que se ha agotado la memoria del microordenador; al final han sido utilizadas hasta zonas de memoria que el micro reserva para sus propias funciones, concretamente, la que reserva para controlar la impresora.

El programa puede ser adaptado para estudiar otras zonas geográficas sin más que cambiar la pantalla del mapa y los datos.

Se incluye una subrutina que permite la cómoda modificación de los datos para su actualización. Esta subrutina permite también, cambiar, con relativa facilidad, un tipo de datos por otros nuevos que se quieran introducir.

Durante la VII Semana de Extremadura en la Escuela se situaron ordenadores con este programa en algunos locales de diversos pueblos de la región, para que fueran utilizados libremente por el público (para la mayoría este fue su primer contacto con un microordenador). Una dificultad que apareció con cierta frecuencia consistía en que algunos usuarios iniciados en la utilización de microordenadores, hacían "Break" para pedir un listado y, cuando terminaban de curiosear, pulsaban "Run". Esto originaba el borrado de las variables y el programa dejaba de funcionar. Para evitar esta situación se chequean todas las teclas que van siendo pulsadas y se ignoran las que pueden producir errores, y se ha añadido una subrutina que impide que el programa sea abierto.

Las personas que deseen una copia en cinta de *cassette* de este programa puede pedirlo a la Sección de Informática del ICE de la Universidad de Extremadura, Avda. de Elvas, s/n. Badajoz.

Por E. Gómez, A. Pérez y M. Pérez. Sección de informática del ICE de la Universidad de Extremadura.

Desensamblador

Imagínese que después de invertir su dinero en un «maravilloso programa» para su Spectrum descubre al llegar a casa que el preciado *software* no hace exactamente lo que se decía o quizá lo que usted pensaba que haría. O imagínese que desea hacer una modificación a una rutina de código máquina y se encuentra con que el fichero que contenía el programa fuente, da insistentemente error de carga.

En cualquiera de estos casos se verá a sí mismo estudiando una larguísima lista de códigos hexadecimales, tratando de discernir qué hace el dichoso programa. Y aunque el mismo manual de su Spectrum contiene una lista de nemónicos y códigos asociados, no es una tarea sencilla ni agradable; dejando aparte el agravante de que hay instrucciones de hasta 4 *bytes* de longitud que dificultan considerablemente el «desensamblado manual».

Este es el problema que el programa propuesto resolverá. Precisamente la interpretación de listas de códigos y otros trabajos mecánicos de esta índole son tareas en las cuales los ordenadores no tienen rival posible. Esto libera al programador permitiéndole emplear su tiempo en tareas más creativas.

Quizá los profanos en materia de lenguaje máquina sigan sin comprender aún la necesidad de tal herramienta. Para ellos daremos una breve explicación al mínimo nivel posible.

Los fabricantes de microprocesadores crean, al diseñar uno de estos elementos, un lenguaje de programación formado por comandos simples que controlan el microprocesador, llamado lenguaje máquina. El lenguaje máquina propiamente dicho se compone de ceros y unos agrupados en bloques de 8 *bits* que forman los *bytes*. Cada uno de estos *bytes* tiene un significado propio como comando de

```
   1 LET n$="NOP": RETURN
   2 LET n$="LD BC,"+STR$ FN n(P
EEK (cont+1),PEEK (cont+2)): LET
 cont=cont+2: RETURN
   3 LET n$="LD (BC),A": RETURN
   4 LET n$="INC BC": RETURN
   5 LET n$="INC B": RETURN
   6 LET n$="DEC B": RETURN
   7 LET n$="LD B,"+STR$ (PEEK (
cont+1)): LET cont=cont+1: RETUR
N
   8 LET n$="RLCA": RETURN
   9 LET n$="EX AF,AF'": RETURN
  10 LET n$="ADD HL,BC": RETURN
  11 LET n$="LD A,(BC)": RETURN
  12 LET n$="DEC BC": RETURN
  13 LET n$="INC C": RETURN
  14 LET n$="DEC C": RETURN
  15 LET n$="LD C,"+STR$ (PEEK (
cont+1)): LET cont=cont+1: RETUR
N
  16 LET n$="RRCA": RETURN
  17 LET n$="DJNZ "+STR$ FN j(co
nt): LET cont=cont+1: RETURN
  18 LET n$="LD DE,"+STR$ FN n(P
EEK (cont+1),PEEK (cont+2)): LET
 cont=cont+2: RETURN
  19 LET n$="LD (DE),A": RETURN
  20 LET n$="INC DE": RETURN
  21 LET n$="INC D": RETURN
  22 LET n$="DEC D": RETURN
  23 LET n$="LD D,"+STR$ (PEEK (
cont+1)): LET cont=cont+1: RETUR
N
  24 LET n$="RLA": RETURN
  25 LET n$="JR "+STR$ FN j(cont
): LET cont=cont+1: RETURN
  26 LET n$="ADD HL,DE": RETURN
  27 LET n$="LD A,(DE)": RETURN
  28 LET n$="DEC DE": RETURN
  29 LET n$="INC E": RETURN
  30 LET n$="DEC E": RETURN
  31 LET n$="LD E,"+STR$ (PEEK (
cont+1)): LET cont=cont+1: RETUR
N
  32 LET n$="RRA": RETURN
  33 LET n$="JR NZ,"+STR$ FN j(c
ont): LET cont=cont+1: RETURN
  34 LET n$="LD HL,"+STR$ FN n(P
EEK (cont+1),PEEK (cont+2)): LET
```

```
 cont=cont+2: RETURN
  35 LET n$="LD ("+STR$ FN n(PEE
K (cont+1),PEEK (cont+2))+"),HL"
: LET cont=cont+2: RETURN
  36 LET n$="INC HL": RETURN
  37 LET n$="INC H": RETURN
  38 LET n$="DEC H": RETURN
  39 LET n$="LD H,"+STR$ (PEEK (
cont+1)): LET cont=cont+1: RETUR
N
  40 LET n$="DAA": RETURN
  41 LET n$="JR Z,"+STR$ FN j(co
nt): LET cont=cont+1: RETURN
  42 LET n$="ADD HL,HL": RETURN
  43 LET n$="LD HL,("+STR$ FN n(
PEEK (cont+1),PEEK (cont+2))+")"
: LET cont=cont+2: RETURN
  44 LET n$="DEC HL": RETURN
  45 LET n$="INC L": RETURN
  46 LET n$="DEC L": RETURN
  47 LET n$="LD L,"+STR$ (PEEK (
cont+1)): LET cont=cont+1: RETUR
N
  48 LET n$="CPL": RETURN
  49 LET n$="JR NC,"+STR$ FN j(c
ont): LET cont=cont+1: RETURN
  50 LET n$="LD SP,"+STR$ FN n(P
EEK (cont+1),PEEK (cont+2)): LET
 cont=cont+2: RETURN
  51 LET n$="LD ("+STR$ FN n(PEE
K (cont+1),PEEK (cont+2))+"),A":
 LET cont=cont+2: RETURN
  52 LET n$="INC SP": RETURN
  53 LET n$="INC (HL)": RETURN
  54 LET n$="DEC (HL)": RETURN
  55 LET n$="LD (HL),"+STR$ (PEE
K (cont+1)): LET cont=cont+1: RE
TURN
  56 LET n$="SCF": RETURN
  57 LET n$="JR C,"+STR$ FN j(co
nt): LET cont=cont+1: RETURN
  58 LET n$="ADD HL,SP": RETURN
  59 LET n$="LD A,("+STR$ FN n(P
EEK (cont+1),PEEK (cont+2))+")":
 LET cont=cont+2: RETURN
  60 LET n$="DEC SP": RETURN
  61 LET n$="INC A": RETURN
  62 LET n$="DEC A": RETURN
  63 LET n$="LD A,"+STR$ (PEEK (
cont+1)): LET cont=cont+1: RETUR
```

```
  64 LET n$="CCF": RETURN
  65 LET n$="LD B,B": RETURN
  66 LET n$="LD B,C": RETURN
  67 LET n$="LD B,D": RETURN
  68 LET n$="LD B,E": RETURN
  69 LET n$="LD B,H": RETURN
  70 LET n$="LD B,L": RETURN
  71 LET n$="LD B,(HL)": RETURN
  72 LET n$="LD B,A": RETURN
  73 LET n$="LD C,B": RETURN
  74 LET n$="LD C,C": RETURN
  75 LET n$="LD C,D": RETURN
  76 LET n$="LD C,E": RETURN
  77 LET n$="LD C,H": RETURN
  78 LET n$="LD C,L": RETURN
  79 LET n$="LD C,(HL)": RETURN
  80 LET n$="LD C,A": RETURN
  81 LET n$="LD D,B": RETURN
  82 LET n$="LD D,C": RETURN
  83 LET n$="LD D,D": RETURN
  84 LET n$="LD D,E": RETURN
  85 LET n$="LD D,H": RETURN
  86 LET n$="LD D,L": RETURN
  87 LET n$="LD D,(HL)": RETURN
  88 LET n$="LD D,A": RETURN
  89 LET n$="LD E,B": RETURN
  90 LET n$="LD E,C": RETURN
  91 LET n$="LD E,D": RETURN
  92 LET n$="LD E,E": RETURN
  93 LET n$="LD E,H": RETURN
  94 LET n$="LD E,L": RETURN
  95 LET n$="LD E,(HL)": RETURN
  96 LET n$="LD E,A": RETURN
  97 LET n$="LD H,B": RETURN
  98 LET n$="LD H,C": RETURN
  99 LET n$="LD H,D": RETURN
 100 LET n$="LD H,E": RETURN
 101 LET n$="LD H,H": RETURN
 102 LET n$="LD H,L": RETURN
 103 LET n$="LD H,(HL)": RETURN
 104 LET n$="LD H,A": RETURN
 105 LET n$="LD L,B": RETURN
 106 LET n$="LD L,C": RETURN
 107 LET n$="LD L,D": RETURN
 108 LET n$="LD L,E": RETURN
 109 LET n$="LD L,H": RETURN
 110 LET n$="LD L,L": RETURN
 111 LET n$="LD L,(HL)": RETURN
 112 LET n$="LD L,A": RETURN
 113 LET n$="LD (HL),B": RETURN
```

del Z80

Con frecuencia oímos hablar de ensambladores-desensambladores, y de sus ventajas en la programación del código máquina. Javier Cancela nos desvela los misterios de estos últimos, y el listado completo del programa.



ejecución de una orden o bien como una dirección de memoria o cifra. Para hacer comprensibles estos códigos se asocia a cada uno de ellos un nombre o nemónico que describe brevemente su función y sus argumentos, ya que de otra forma sería prácticamente imposible realizar programas de importancia en el lenguaje madre de los microprocesadores. Existen programas

```
114 LET n$="LD (HL),C": RETURN
115 LET n$="LD (HL),D": RETURN
116 LET n$="LD (HL),E": RETURN
117 LET n$="LD (HL),H": RETURN
118 LET n$="LD (HL),L": RETURN
119 LET n$="HALT": RETURN
120 LET n$="LD (HL),A": RETURN
121 LET n$="LD A,B": RETURN
122 LET n$="LD A,C": RETURN
123 LET n$="LD A,D": RETURN
124 LET n$="LD A,E": RETURN
125 LET n$="LD A,H": RETURN
126 LET n$="LD A,L": RETURN
127 LET n$="LD A,(HL)": RETURN
128 LET n$="LD A,A": RETURN
129 LET n$="ADD A,B": RETURN
130 LET n$="ADD A,C": RETURN
131 LET n$="ADD A,D": RETURN
132 LET n$="ADD A,E": RETURN
133 LET n$="ADD A,H": RETURN
134 LET n$="ADD A,L": RETURN
135 LET n$="ADD A,(HL)": RETURN

136 LET n$="ADD A,A": RETURN
137 LET n$="ADC A,B": RETURN
138 LET n$="ADC A,C": RETURN
139 LET n$="ADC A,D": RETURN
140 LET n$="ADC A,E": RETURN
141 LET n$="ADC A,H": RETURN
142 LET n$="ADC A,L": RETURN
143 LET n$="ADC A,(HL)": RETURN

144 LET n$="ADC A,A": RETURN
145 LET n$="SUB B": RETURN
146 LET n$="SUB C": RETURN
147 LET n$="SUB D": RETURN
148 LET n$="SUB E": RETURN
149 LET n$="SUB H": RETURN
150 LET n$="SUB L": RETURN
151 LET n$="SUB (HL)": RETURN
152 LET n$="SUB A": RETURN
153 LET n$="SBC A,B": RETURN
154 LET n$="SBC A,C": RETURN
155 LET n$="SBC A,D": RETURN
156 LET n$="SBC A,E": RETURN
157 LET n$="SBC A,H": RETURN
158 LET n$="SBC A,L": RETURN
159 LET n$="SBC A,(HL)": RETURN
160 LET n$="SBC A,A": RETURN
161 LET n$="AND B": RETURN
```

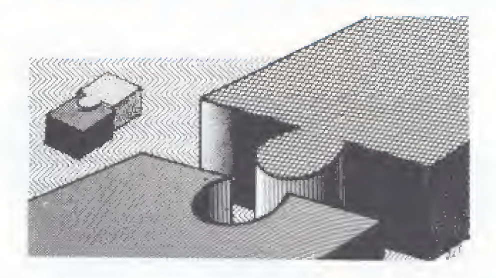
```
162 LET n$="AND C": RETURN
163 LET n$="AND D": RETURN
164 LET n$="AND E": RETURN
165 LET n$="AND H": RETURN
166 LET n$="AND L": RETURN
167 LET n$="AND (HL)": RETURN
168 LET n$="AND A": RETURN
169 LET n$="XOR B": RETURN
170 LET n$="XOR C": RETURN
171 LET n$="XOR D": RETURN
172 LET n$="XOR E": RETURN
173 LET n$="XOR H": RETURN
174 LET n$="xOR L": RETURN
175 LET n$="XOR (HL)": RETURN
176 LET n$="XOR A": RETURN
177 LET n$="OR B": RETURN
178 LET n$="OR C": RETURN
179 LET n$="OR D": RETURN
180 LET n$="OR E": RETURN
181 LET n$="OR H": RETURN
182 LET n$="OR L": RETURN
183 LET n$="OR (HL)": RETURN
184 LET n$="OR A": RETURN
185 LET n$="CP B": RETURN
186 LET n$="CP C": RETURN
187 LET n$="CP D": RETURN
188 LET n$="CP E": RETURN
189 LET n$="CP H": RETURN
190 LET n$="CP L": RETURN
191 LET n$="CP (HL)": RETURN
192 LET n$="CP A": RETURN
193 LET n$="RET NZ": RETURN
194 LET n$="POP BC": RETURN
195 LET n$="JP NZ,"+STR$ FN n(P
EEK (cont+1),PEEK (cont+2)): LET
cont=cont+2: RETURN
196 LET n$="JP "+STR$ FN n(PEEK
(cont+1),PEEK (cont+2)): LET co
nt=cont+2: RETURN
197 LET n$="CALL NZ,"+STR$ FN n
(PEEK (cont+1),PEEK (cont+2)): L
ET cont=cont+2: RETURN
198 LET n$="PUSH BC": RETURN
199 LET n$="ADD A,"+STR$ (PEEK
(cont+1)): LET cont=cont+1: RETU
RN
200 LET n$="RST 0": RETURN
201 LET n$="RET Z": RETURN
202 LET n$="RET": RETURN
203 LET n$="JP Z,"+STR$ FN n(PE
```

```
EK (cont+1),PEEK (cont+2)): LET
cont=cont+2: RETURN
204 LET cont=cont+1: GO SUB (30
0+PEEK cont): RETURN
205 LET n$="CALL Z,"+STR$ FN n(
PEEK (cont+1),PEEK (cont+2)): LE
T cont=cont+2: RETURN
206 LET n$="CALL "+STR$ FN n(PE
EK (cont+1),PEEK (cont+2)): LET
cont=cont+2: RETURN
207 LET n$="ADC A,"+STR$ (PEEK
(cont+1)): LET cont=cont+1: RETU
RN
208 LET n$="RST 8": RETURN
209 LET n$="RET NC": RETURN
210 LET n$="POP DE": RETURN
211 LET n$="JP NC,"+STR$ FN n(P
EEK (cont+1),PEEK (cont+2)): LET
cont=cont+2: RETURN
212 LET n$="OUT ("+STR$ (PEEK (
cont+1))+"),A": LET cont=cont+1:
RETURN
213 LET n$="CALL NC,"+STR$ FN n
(PEEK (cont+1),PEEK (cont+2)): L
ET cont=cont+2: RETURN
214 LET n$="PUSH DE": RETURN
215 LET n$="SUB "+STR$ (PEEK (c
ont+1)): LET cont=cont+1: RETURN

216 LET n$="RST 16": RETURN
217 LET n$="RET C": RETURN
218 LET n$="EXX": RETURN
219 LET n$="JP C,"+STR$ FN n(PE
EK (cont+1),PEEK (cont+2)): LET
cont=cont+2: RETURN
220 LET n$="IN A,("+STR$ (PEEK
(cont+1))+")": LET cont=cont+1:
RETURN
221 LET n$="CALL C,"+STR$ FN n(
PEEK (cont+1),PEEK (cont+2)): LE
T cont=cont+2: RETURN
222 LET cont=cont+1: LET i$="IX
": GO SUB (900+PEEK cont): RETUR
N
223 LET n$="SBC A,"+STR$ (PEEK
(cont+1)): LET cont=cont+1: RETU
RN
224 LET n$="RST 24": RETURN
225 LET n$="RET PO": RETURN
226 LET n$="POP HL": RETURN
```

llamados ensambladores que generan a partir de un texto de nemónicos los *bytes* que forman el programa en máquina en la memoria del ordenador. Los desensambladores realizan precisamente la función contraria: generan el texto de nemónicos a partir de los códigos.

Después de esta pequeña clase, nos centraremos en el programa en cuestión. Está realizado en BASIC y es de gran sencillez para facilitar la comprensión de su funcionamiento. El precio de ello es una mayor longitud y por tanto más memoria ocupada. Optamos por esta solución en lugar de un programa más corto, pero con técnicas que harían notablemente más difícil su entendimiento. El programa puede desensamblar código en cualquier dirección de memoria que no pertenezca a la porción ocupada por el propio programa en BASIC. El *software* a procesar que se halle en esta zona de memoria podrá ser decodificado simplemente cargándolo en una dirección que se halle libre; con la facilidad que para ello nos da el comando del Spectrum LOAD



""CODE nnnnn; donde nnnnn es la dirección de memoria. El programa preguntará dónde se halla realmente el código *(dirección real)* y dónde se debería hallar *(dirección virtual)*: de forma que el desensamblado aparecerá en pantalla con las direcciones y saltos correspondientes a la zona donde normalmente reside el código, y donde verdaderamente debe ejecutarse. Huelga decir que el programa a desensamblar deberá cargarse siempre por encima del RAMTOP establecido si no se quiere corromper el propio desensamblador. El RAMTOP será determinado por el propio usuario antes de cargar el programa BASIC, de forma que quede espacio suficiente para el programa en máquina. El CLEAR más bajo que se puede realizar con el desensamblador cargado es de aproximadamente 40000, ya que el programa ocupa unos 15 *kbytes* de memoria, que se convierten en 16 con el espacio ocupado por las variables y el sistema BASIC.

El programa dará un desensamblado de cualquier código, pero si éste es una tabla de datos, o una cadena de caracteres, el resultado será un algoritmo sin sentido; ya que no puede ser capaz de diferenciar datos de verdaderas instrucciones.

El funcionamiento del programa es de una simplicidad absoluta: al ser cargado se pondrá en funcionamiento e inmediatamente preguntará la dirección real, o el lugar donde se encuentra realmente el programa, y posteriormente pedirá la dirección donde debería residir (obviamente se deberá contestar a las dos preguntas con la misma respuesta si no se ha desplazado el código de su dirección de ejecución). Después se deberá proporcionar el número de *bytes* que se desean desensamblar. Realizando el desensamblado en varios bloques se pueden evitar tablas o cadenas que

```
  227 LET n$="JP PO,"+STR$ FN n(P
EEK (cont+1),PEEK (cont+2)): LET
 cont=cont+2: RETURN
  228 LET n$="EX (SP),HL": RETURN

  229 LET n$="CALL PO,"+STR$ FN n
(PEEK (cont+1),PEEK (cont+2)): L
ET cont=cont+2: RETURN
  230 LET n$="PUSH HL": RETURN
  231 LET n$="AND "+STR$ (PEEK (c
ont+1)): LET cont=cont+1: RETURN

  232 LET n$="RST 32": RETURN
  233 LET n$="RET PE": RETURN
  234 LET n$="JP (HL)": RETURN
  235 LET n$="JP PE,"+STR$ FN n(P
EEK (cont+1),PEEK (cont+2)): LET
 cont=cont+2: RETURN
  236 LET n$="EX DE,HL": RETURN
  237 LET n$="CALL PE,"+STR$ FN n
(PEEK (cont+1),PEEK (cont+2)): L
ET cont=cont+2: RETURN
  238 LET cont=cont+1: GO SUB (60
0+PEEK cont): RETURN
  239 LET n$="XOR "+STR$ (PEEK (c
ont+1)): LET cont=cont+1: RETURN
  240 LET n$="RST 40": RETURN
  241 LET n$="RET P": RETURN
  242 LET n$="POP AF": RETURN
  243 LET n$="JP P,"+STR$ FN n(PE
EK (cont+1),PEEK (cont+2)): LET
cont=cont+2: RETURN
  244 LET n$="DI": RETURN
  245 LET n$="CALL P,"+STR$ FN n(
PEEK (cont+1),PEEK (cont+2)): LE
T cont=cont+2: RETURN
  246 LET n$="PUSH AF": RETURN
  247 LET n$="OR "+STR$ (PEEK (co
nt+1)): LET cont=cont+1: RETURN
  248 LET n$="RST 48": RETURN
  249 LET n$="RET M": RETURN
```

```
  250 LET n$="LD SP,HL": RETURN
  251 LET n$="JP M,"+STR$ FN n(PE
EK (cont+1),PEEK (cont+2)): LET
cont=cont+2: RETURN
  252 LET n$="EI": RETURN
  253 LET n$="CALL M,"+STR$ FN n(
PEEK (cont+1),PEEK (cont+2)): LE
T cont=cont+2: RETURN
  254 LET cont=cont+1: LET i$="IY
": GO SUB (900+PEEK cont): RETUR
N
  255 LET n$="CP "+STR$ (PEEK (co
nt+1)): LET cont=cont+1: RETURN
  256 LET n$="RST 56": RETURN
  300 LET n$="RLC B": RETURN
  301 LET n$="RLC C": RETURN
  302 LET n$="RLC D": RETURN
  303 LET n$="RLC E": RETURN
  304 LET n$="RLC H": RETURN
  305 LET n$="RLC L": RETURN
  306 LET n$="RLC (HL)": RETURN
  307 LET n$="RLC A": RETURN
  308 LET n$="RRC B": RETURN
  309 LET n$="RRC C": RETURN
  310 LET n$="RRC D": RETURN
  311 LET n$="RRC E": RETURN
  312 LET n$="RRC H": RETURN
  313 LET n$="RRC L": RETURN
  314 LET n$="RRC (HL)": RETURN
  315 LET n$="RRC A": RETURN
  316 LET n$="RL B": RETURN
  317 LET n$="RL C": RETURN
  318 LET n$="RL D": RETURN
  319 LET n$="RL E": RETURN
  320 LET n$="RL H": RETURN
  321 LET n$="RL L": RETURN
  322 LET n$="RL (HL)": RETURN
  323 LET n$="RL A": RETURN
  324 LET n$="RR B": RETURN
  325 LET n$="RR C": RETURN
  326 LET n$="RR D": RETURN
```

```
  327 LET n$="RR E": RETURN
  328 LET n$="RR H": RETURN
  329 LET n$="RR L": RETURN
  330 LET n$="RR (HL)": RETURN
  331 LET n$="RR A": RETURN
  332 LET n$="SLA B": RETURN
  333 LET n$="SLA C": RETURN
  334 LET n$="SLA D": RETURN
  335 LET n$="SLA E": RETURN
  336 LET n$="SLA H": RETURN
  337 LET n$="SLA L": RETURN
  338 LET n$="SLA (HL)": RETURN
  339 LET n$="SLA A": RETURN
  340 LET n$="SRA B": RETURN
  341 LET n$="SRA C": RETURN
  342 LET n$="SRA D": RETURN
  343 LET n$="SRA E": RETURN
  344 LET n$="SRA H": RETURN
  345 LET n$="SRA L": RETURN
  346 LET n$="SRA (HL)": RETURN
  347 LET n$="SRA A": RETURN
  355 LET n$="Error": RETURN
  356 LET n$="SRL B": RETURN
  357 LET n$="SRL C": RETURN
  358 LET n$="SRL D": RETURN
  359 LET n$="SRL E": RETURN
  360 LET n$="SRL H": RETURN
  361 LET n$="SRL L": RETURN
  362 LET n$="SRL (HL)": RETURN
  363 LET n$="SRL A": RETURN
  371 LET n$="BIT 0,": GO SUB 500
0: RETURN
  379 LET n$="BIT 1,": GO SUB 500
0: RETURN
  387 LET n$="BIT 2,": GO SUB 500
0: RETURN
  395 LET n$="BIT 3,": GO SUB 500
0: RETURN
  403 LET n$="BIT 4,": GO SUB 500
0: RETURN
  411 LET n$="BIT 5,": GO SUB 500
```

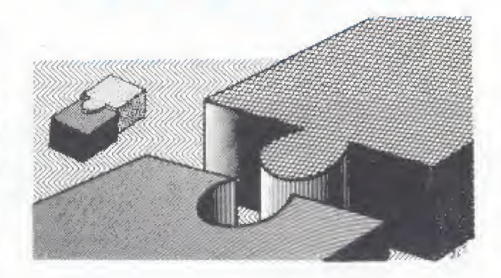


Programas/Juegos/Montajes/Código Máquina
AÑO II - Núm. 19 - Junio 1985 - 250 Ptas.
ZX
REVISTA PARA LOS USUARIOS DE ORDENADORES SINCLAIR
EL SOFTWARE QUE NOS INVADE
Analizamos las revistas en cassette
Iteraciones con el ZX-81
Scroll de ventanas
¡GANE ESTA VESPA!
YA ESTA A LA VENTA
Cómo jugar al Knight Lore

z80

se hallen entre medias del programa en código máquina. Por último será necesario contestar si se desea volcar el desensamblado por una impresora o bien por pantalla. La opción de impresora está pensada en principio para impresoras de 80 columnas, de forma que se aproveche el espacio extra, sacando dos instrucciones de programa por cada línea. Para aquellos usuarios más modestos que sólo dispongan de la impresora de Sinclair, explicaremos más adelante cómo modificar el programa para usar este periférico.

En cuanto a los listados resultantes hemos de especificar que todas las cifras aparecen en decimal. Escogimos esta alternativa en lugar del sistema hexadecimal (mucho más extendido en el mundo del lenguaje máquina), debido a que el Spectrum no dispone de la posibilidad de escribir números en



esta base, ni de funciones de conversión de tipo alguno; y esto ha llevado a que la mayoría de los usuarios estemos acostumbrados a manejar direcciones y *bytes* de memoria en nuestro viejo sistema decimal. Los nemónicos son por supuesto los establecidos por ZILOG, fabricante del procesador Z80. Los saltos relativos de las instrucciones JR y DJNZ son dados en forma de etiquetas; es decir, en lugar del número de *bytes* del salto, aparecerá la dirección de la instrucción a la que se realiza el salto.

Cómo funciona

El programa se empieza a ejecutar en la línea 2000. Como decíamos anteriormente, no se hace ningún CLEAR en él; esta instrucción se debe ejecutar antes que el programa y según la dirección del código. La primera línea fija los colores del papel y la tinta y limpia la pantalla. Por supuesto, se pueden cambiar los colores por otros cualesquiera. Después tenemos dos definiciones de funciones: la primera convierte un salto relativo de una instrucción JR o DJNZ en una etiqueta que indica la dirección donde se realiza el salto. La segunda convierte los valores entre 0 y 255 de dos *bytes* que representen una dirección de memoria en una cifra entre 0 y 65535 (0 a 64 *kbytes*).

Las líneas 2010 a 2025 realizan la entrada de datos necesarios para

```
0: RETURN
 419 LET n$="BIT 6,": GO SUB 500
0: RETURN
 427 LET n$="BIT 7,": GO SUB 500
0: RETURN
 435 LET n$="RES 0,": GO SUB 500
0: RETURN
 443 LET n$="RES 1,": GO SUB 500
0: RETURN
 451 LET n$="RES 2,": GO SUB 500
0: RETURN
 459 LET n$="RES 3,": GO SUB 500
0: RETURN
 467 LET n$="RES 4,": GO SUB 500
0: RETURN
 475 LET n$="RES 5,": GO SUB 500
0: RETURN
 483 LET n$="RES 6,": GO SUB 500
0: RETURN
 491 LET n$="RES 7,": GO SUB 500
0: RETURN
 499 LET n$="SET 0,": GO SUB 500
0: RETURN
 507 LET n$="SET 1,": GO SUB 500
0: RETURN
 515 LET n$="SET 2,": GO SUB 500
0: RETURN
 523 LET n$="SET 3,": GO SUB 500
0: RETURN
 531 LET n$="SET 4,": GO SUB 500
0: RETURN
 539 LET n$="SET 5,": GO SUB 500
0: RETURN
 547 LET n$="SET 6,": GO SUB 500
0: RETURN
 555 LET n$="SET 7,": GO SUB 500
0: RETURN
 664 LET n$="IN B,(C)": RETURN
 665 LET n$="OUT (C),B": RETURN
 666 LET n$="SBC HL,BC": RETURN
 667 LET n$="LD ("+STR$ FN n(PEE
K (cont+1),PEEK (cont+2))+"),BC"
: LET cont=cont+2: RETURN
 668 LET n$="NEG": RETURN
 669 LET n$="RETN": RETURN
 670 LET n$="IM 0": RETURN
 671 LET n$="LD I,A": RETURN
 672 LET n$="IN C,(C)": RETURN
 673 LET n$="OUT C,(C)": RETURN
 674 LET n$="ADC HL,BC": RETURN
```

```
 675 LET n$="LD BC,("+STR$ FN n(
PEEK (cont+1),PEEK (cont+2))+")"
: LET cont=cont+2: RETURN
 677 LET n$="RETI": RETURN
 679 LET n$="LD R,A": RETURN
 680 LET n$="IN D,(C)": RETURN
 681 LET n$="OUT (C),D": RETURN
 682 LET n$="SBC HL,DE": RETURN
 683 LET n$="LD ("+STR$ FN n(PEE
K (cont+1),PEEK (cont+2))+"),DE"
: LET cont=cont+2: RETURN
 686 LET n$="IM 1": RETURN
 687 LET n$="LD A,I": RETURN
 688 LET n$="IN E,(C)": RETURN
 689 LET n$="OUT (C),E": RETURN
 690 LET n$="ADC HL,DE": RETURN
 691 LET n$="LD DE,("+STR$ FN n(
PEEK (cont+1),PEEK (cont+2))+")"
: LET cont=cont+2: RETURN
 694 LET n$="IM 2": RETURN
 695 LET n$="LD A,R": RETURN
 696 LET n$="IN H,(C)": RETURN
 697 LET n$="OUT (C),H": RETURN
 698 LET n$="SBC HL,HL": RETURN
 699 LET n$="LD ("+STR$ FN n(PEE
K (cont+1),PEEK (cont+2))+"),HL"
: LET cont=cont+2: RETURN
 703 LET n$="RRD": RETURN
 704 LET n$="IN L,(C)": RETURN
 705 LET n$="OUT (C),L": RETURN
 706 LET n$="ADC HL,HL": RETURN
 707 LET n$="LD HL,("+STR$ FN n(
PEEK (cont+1),PEEK (cont+2))+")"
: LET cont=cont+2: RETURN
 711 LET n$="RLD": RETURN
 712 LET n$="IN F,(C)": RETURN
 714 LET n$="SBC HL,SP": RETURN
 715 LET n$="LD("+STR$ FN n(PEEK
 (cont+1),PEEK (cont+2))+")SP":
LET cont=cont+2: RETURN
 720 LET n$="IN A,(C)": RETURN
 721 LET n$="OUT (C),A": RETURN
 722 LET n$="ADC HL,SP": RETURN
 723 LET n$="LD SP,("+STR$ FN n(
PEEK (cont+1),PEEK (cont+2))+")"
: LET cont=cont+2: RETURN
 760 LET n$="LDI": RETURN
 761 LET n$="CPI": RETURN
 762 LET n$="INI": RETURN
```

```
 763 LET n$="OUTI": RETURN
 768 LET n$="LDD": RETURN
 769 LET n$="CPD": RETURN
 770 LET n$="IND": RETURN
 771 LET n$="OUTD": RETURN
 776 LET n$="LDIR": RETURN
 777 LET n$="CPIR": RETURN
 778 LET n$="INIR": RETURN
 779 LET n$="OTIR": RETURN
 784 LET n$="LDDR": RETURN
 785 LET n$="CPDR": RETURN
 786 LET n$="INDR": RETURN
 787 LET n$="OTDR": RETURN
 909 LET n$="ADD "+i$+",BC": RET
URN
 925 LET n$="ADD "+i$+",DE": RET
URN
 933 LET n$="LD "+i$+","+STR$ FN
 n(PEEK (cont+1),PEEK (cont+2)):
 LET cont=cont+2: RETURN
 934 LET n$="LD ("+STR$ FN n(PEE
K (cont+1),PEEK (cont+2))+"),"+i
$: LET cont=cont+2: RETURN
 935 LET n$="INC "+i$: RETURN
 941 LET n$="ADD "+i$+","+i$: RE
TURN
 942 LET n$="LD "+i$+",("+STR$ F
N n(PEEK (cont+1),PEEK (cont+2))
+")": LET cont=cont+2: RETURN
 943 LET n$="DEC "+i$: RETURN
 952 LET n$="INC ("+i$+"+"+STR$
PEEK (cont+1)+")": LET cont=cont
+1: RETURN
 953 LET n$="DEC ("+i$+"+"+STR$
PEEK (cont+1)+")": LET cont=cont
+1: RETURN
 954 LET n$="LD ("+i$+"+"+STR$ P
EEK (cont+1)+"),"+STR$ PEEK (con
t+2): LET cont=cont+2: RETURN
 957 LET n$="ADD "+i$+",SP": RET
URN
 970 LET n$="LD B,("+i$+"+"+STR$
 PEEK (cont+1)+")": LET cont=con
t+1: RETURN
 978 LET n$="LD C,("+i$+"+"+STR$
 PEEK (cont+1)+")": LET cont=con
t+1: RETURN
 986 LET n$="LD D,("+i$+"+"+STR$
 PEEK (cont+1)+")": LET cont=con
```

el funcionamiento del programa. La dirección real del programa es almacenada en la variable *«dir»*. La dirección virtual o de ejecución se toma en la variable *«pos»;* y el número de *bytes* es guardado en *«lon»*. La variable *«imp»* tomará el valor 1 cuando se desea salida por impresora. En la línea 2030 se inicializan algunas variables: *«cont»* es el puntero de dirección de la instrucción en curso de decodificación; y *«offset»* es la diferencia entre las direcciones real y virtual, que será usada posteriormente para calcular los saltos relativos, etc. La variable *«col»* es usada para la impresión a 80 columnas, de forma que el carro salte de línea en el momento adecuado.

Las líneas 2040 a 2080 constituyen el bucle principal del programa. En la 2040 se toma el *byte* al que apunta el puntero de dirección. En la 2045 aparece la variable *«d»*, que será la dirección que aparecerá en pantalla, es decir, la dirección real de ejecución del código. Esta se obtiene añadiendo al puntero *«cont»* el desplazamiento *«offset»*. La línea 2050 es realmente la clave del funcionamiento de todo el programa. Esta instrucción llama a una subrutina cuyo número de línea es igual al valor del *byte* de código tomado más uno. De esta forma según la instrucción a decodificar, el programa llamará a una subrutina o a otra. En cada una de las distintas subrutinas (formadas por una sola línea de programa), se asigna un valor a la variable alfanumérica *«n$»*. Esta variable correspone a nemónico de la instrucción decodificada. Por otra parte se calculan los saltos o direcciones que pudiesen existir en la instrucción mediante llamadas a las funciones definidas de las que se habló antes. Si la instrucción está formada por más de 1 *byte,* el puntero es incrementado según corresponda. Una vez realizadas estas funciones, el control es devuelto al bucle principal. En la instrucción 2060 se imprime en pantalla la instrucción decodificada; en primer lugar la dirección donde reside (*«d»*) y en segundo lugar el nemónico con sus operandos (*«n$»*). Si no se ha escogido la opción de impresora se salta el bloque de líneas que realizan esta función.

La línea 2063 «pokea» en la variable del sistema SCR CT un valor mayor de 1, evitando que el ordenador se pare cuando se llene una pantalla preguntando «scroll?». De esta forma, el listado saldrá a la vez por impresora y pantalla, sin necesidad de presionar una tecla cada 22 líneas. Las líneas 2065 y 2067 ejecutan la salida por impresora a 80 columnas. Si se dispone de una impresora Sinclair

```
t+1: RETURN
 994 LET n$="LD E,("+i$+"+"+STR$
 PEEK (cont+1)+")": LET cont=con
t+1: RETURN
1002 LET n$="LD H,("+i$+"+"+STR$
 PEEK (cont+1)+")": LET cont=con
t+1: RETURN
1010 LET n$="LD L,("+i$+"+"+STR$
 PEEK (cont+1)+")": LET cont=con
t+1: RETURN
1012 LET n$="LD ("+i$+"+"+STR$ P
EEK (cont+1)+"),B": LET cont=con
t+1: RETURN
1013 LET n$="LD ("+i$+"+"+STR$ P
EEK (cont+1)+"),C": LET cont=con
t+1: RETURN
1014 LET n$="LD ("+i$+"+"+STR$ P
EEK (cont+1)+"),D": LET cont=con
t+1: RETURN
1015 LET n$="LD ("+i$+"+"+STR$ P
EEK (cont+1)+"),E": LET cont=con
t+1: RETURN
1016 LET n$="LD ("+i$+"+"+STR$ P
EEK (cont+1)+"),H": LET cont=con
t+1: RETURN
1017 LET n$="LD ("+i$+"+"+STR$ P
EEK (cont+1)+"),L":LET cont=con
t+1: RETURN
1019>LET n$="LD ("+i$+"+"+STR$ P
EEK (cont+1)+"),A":LET cont=cont
+1:RETURN
1026 LET n$="LD A,("+i$+"+"+STR$
 PEEK (cont+1)+")": LET cont=con
t+1: RETURN
1034 LET n$="ADD ("+i$+"+"+STR$
PEEK (cont+1)+")": LET cont=cont
+1: RETURN
1042 LET n$="ADC ("+i$+"+"+STR$
PEEK (cont+1)+")": LET cont=cont
+1: RETURN
1050 LET n$="SUB ("+i$+"+"+STR$
PEEK (cont+1)+")": LET cont=cont
+1: RETURN
1058 LET n$="SBC ("+i$+"+"+STR$
PEEK (cont+1)+")": LET cont=cont
+1: RETURN
1066 LET n$="AND ("+i$+"+"+STR$
PEEK (cont+1)+")": LET cont=cont
+1: RETURN
1074 LET n$="XOR ("+i$+"+"+STR$
```

```
PEEK (cont+1)+")": LET cont=cont
+1: RETURN
1082 LET n$="OR ("+i$+"+"+STR$ P
EEK (cont+1)+")": LET cont=cont+
1: RETURN
1090 LET n$="CP ("+i$+"+"+STR$ P
EEK (cont+1)+")": LET cont=cont+
1: RETURN
1103 LET cont=cont+2: GO SUB (15
00+PEEK cont): RETURN
1125 LET n$="POP "+i$: RETURN
1127 LET n$="EX (SP),"+i$: RETUR
N
1129 LET n$="PUSH "+i$: RETURN
1133 LET n$="JP ("+i$+")": RETUR
N
1506 LET n$="RLC (A),("+i$+"+"+S
TR$ PEEK (cont-1)+")": RETURN
1514 LET n$="RRC (A),("+i$+"+"+S
TR$ PEEK (cont-1)+")": RETURN
1522 LET n$="RL (A),("+i$+"+"+ST
R$ PEEK (cont-1)+")": RETURN
1530 LET n$="RR (A),("+i$+"+"+ST
R$ PEEK (cont-1)+")": RETURN
1538 LET n$="SLA ("+i$+"+"+STR$
PEEK (cont-1)+")": RETURN
1546 LET n$="SRA ("+i$+"+"+STR$
PEEK (cont-1)+")": RETURN
1562 LET n$="SRL ("+i$+"+"+STR$
PEEK (cont-1)+")": RETURN
1626 LET n$="BIT "+STR$ ((PEEK (
cont)-70)/8)+",("+i$+"+"+STR$ PE
EK (cont-1)+")": RETURN
1690 LET n$="RES "+STR$ ((PEEK (
cont)-134)/8)+",("+i$+"+"+STR$ P
EEK (cont-1)+")": RETURN
1754 LET n$="SET "+STR$ ((PEEK (
cont)-198)/8)+",("+i$+"+"+STR$ P
EEK (cont-1)+")": RETURN
2000 INK 7: PAPER 0: BORDER 0: C
LS
2001 DEF FN j(c)=(PEEK (c+1)+((P
EEK (c+1)<=127)*(c+2))-((PEEK (c
+1)>127)*(PEEK (c+1)))+((PEEK (c
+1)>127)*(c-254+PEEK (c+1))))+of
fset
2005 DEF FN n(l,h)=256*h+l
2010 INPUT "Direccion real del p
rograma?",dir
2015 INPUT "Direccion virtual de
```

```
l programa?",pos
2020 INPUT "Numero de bytes?";lo
n
2025 LET imp=0: INPUT "Impresora
? (s/n)";x$: IF x$="s" THEN LET
 imp=1: LPRINT CHR$ 13
2030 LET cont=dir: LET offset=po
s-dir: LET col=0
2040 LET byte=PEEK cont: LET col
=col+1
2045 LET d=cont+offset
2050 GO SUB byte+1
2060 PRINT d,n$: IF imp=0 THEN
GO TO 2070
2063 POKE 23692,255
2065 IF (INT (col/2))<>(col/2) T
HEN LPRINT d,n$,
2067 IF (INT (col/2))=(col/2) TH
EN LPRINT d,n$,,
2070 LET cont=cont+1
2080 IF cont<=(dir+lon-1) THEN
GO TO 2040
2085 IF imp=1 THEN LPRINT CHR$
13
2090 PAUSE 0: GO TO 2000
5000 IF INT (10*(((PEEK cont)/8)
-INT ((PEEK cont)/8)))=0 THEN L
ET n$=n$+"B"
5010 IF INT (10*(((PEEK cont)/8)
-INT ((PEEK cont)/8)))=1 THEN L
ET n$=n$+"C"
5020 IF INT (10*(((PEEK cont)/8)
-INT ((PEEK cont)/8)))=2 THEN L
ET n$=n$+"D"
5030 IF INT (10*(((PEEK cont)/8)
-INT ((PEEK cont)/8)))=3 THEN L
ET n$=n$+"E"
5040 IF INT (10*(((PEEK cont)/8)
-INT ((PEEK cont)/8)))=5 THEN L
ET n$=n$+"H"
5050 IF INT (10*(((PEEK cont)/8)
-INT ((PEEK cont)/8)))=6 THEN L
ET n$=n$+"L"
5060 IF INT (10*(((PEEK cont)/8)
-INT ((PEEK cont)/8)))=7 THEN L
ET n$=n$+"(HL)"
5070 IF INT (10*(((PEEK cont)/8)
-INT ((PEEK cont)/8)))=8 THEN L
ET n$=n$+"A"
5080 RETURN
```

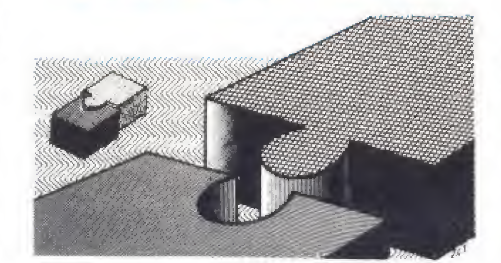
u otra de 32 caracteres por línea, se deben sustituir estas dos instrucciones por esta otra:

2065 LPRINT d,n$

de forma que se imprima un solo nemónico por línea.

En la línea 2070 se incrementa el puntero de dirección. En la 2080 se retorna al principio del bucle y se repite el proceso, siempre y cuando no se haya llegado a decodificar el número de instrucciones requeridas. Si se alcanza esta condición el programa sale del bucle y se envía un carácter de retorno de carro a la impresora (si se escogió esta opción) para impedir la línea que pueda quedar en el *buffer* de ésta. Por último, el programa se detiene hasta que se presiona una tecla y se reinicializa para un nuevo desensamblado.

El resto del programa está formado por las subrutinas de todos y cada uno de los nemónicos que constituyen el lenguaje máquina del Z80. Las líneas 1 a 255 forman



el bloque de instrucciones cuyos códigos de operación están formados por un solo *byte* (la instrucción completa con operandos puede tener más de un *byte*). Las líneas 300 a 555 son las instrucciones cuyo código de operación empieza con el *byte* 203. Lo mismo es aplciable a las líneas 664 a 787, pero con el *byte* 237. Estas subrutinas son llamadas desde las líneas 204 y 238 respectivamente. Las líneas comprendidas entre la 909 y la 1754 son las correspondientes a las instrucciones que manejan los registros índice IX e IY. La rutina a partir de la línea 5000 en adelante, determina el registro con que opera un grupo de instrucciones según el valor de sus códigos, y es llamada desde las subrutinas correspondientes a cada instrucción.

Después de esta profunda descripción del programa, daremos algunas sugerencias para su implementación y utilización. En primer lugar, la importancia fundamental de los NUMEROS de LINEA de las sentencias del programa: no se deben cambiar o alterar, pues en ellos se basa el funcionamiento del programa. Tecleado todo el programa en el ordenador, se salvará con un nombre cualquiera y con el comando LINE 2000, de forma que cada vez que se cargue, comience a ejecutarse automáticamente en la sentencia correcta. Si por cualquier circunstancia se parase la ejecución, se deberá emplear el comando RUN 2000 o GOTO 2000 para recomenzar el programa.

SOFTWARE EDUCATIVO: ¡QUE DIVERTIDO!

Hablar de programas educativos no es sinónimo de programas aburridos. Los que más han estudiado el tema son las casas de software americanas, quienes incorporan en sus plantillas un amplio número de psicólogos y pedagogos. Hablamos con Damià García, gerente de Idealogic y gran conocedor de la problemática educativa de este país.



El ordenador me permite trabajar con el conocimiento y manipular ideas de una manera más rápida y organizada.

En Barcelona, en unos bajos recién estrenados, vamos a visitar a Damià García; gerente de **Idealogic,** empresa editora de *software* educativo.

Recorremos los pasillos del cuartel general. Vemos en una acristalada sala a la derecha que varios programadores trabajan febrilmente. Un cartel lo dice todo: «Sala de Pensar».

Damià nos recibe en su despacho. El es sicólogo de profesión, uno de los pioneros en trabajar con Logo en Barcelona, e impulsor de la edición de programas educativos para micrordenadores.

Le preguntamos el por qué del *software* educativo.

—La idea de por qué debe existir un *soft* educativo para mí, está en la esencia de por qué deben existir ordenadores. El invertir un dinero en un ordenador no es simplemente para tener un juguete más. Uno ordenador debe ser una herramienta de trabajo, una herramienta de culturización, y para aprovechar unos recursos que están ahora a nuestro abasto. Estos son los recursos que nos proporciona la informática. El ordenador es aquel aparato que me permite trabajar con el conocimiento y manipular ideas de una manera mucho más rápida y organizada. Si utilizo un ordenador en su aspecto puramente lúdico, creo que estoy tirando un dinero y dejando de lado las posibilidades de futuro y de aprovechamiento cultural e intelectual que potencialmente tengo ante mí. Esto quiere decir que debe existir un *soft* formativo, didáctico, en última instancia educativo.

No descarto la existencia de otro tipo de *soft;* pero es una lástima que la pasión que tienen los chavales por los ordenadores sólo quede reducida a unas pocas instancias. Toda esta motivación puede permitir una ayuda en la transmisión de conocimientos.

—Asistimos desde hace un par de años a un impresionante boom de usuarios de micrordenadores. Un importantísimo porcentaje de éstos, son niños y jóvenes en edad escolar. Toda su gran afición la canalizan en el consumo de juegos y su mayor interés radica en coleccionarlos.

Han proliferado clubes, revistas y casas dedicadas a la importación de estos programas fundamentalmente del Reino Unido. Sin embargo, este mismo fenómeno aún no se ha desarrollado en toda su extensión en las escuelas. Se han dado pasos importantes, eso sí. Congresos y jornadas sobre temas de educación e informática, un proyecto gubernamental —el proyecto Atenea— y las escuelas privadas más activas ya han incorporado la informática en alguna medida.

Los chavales no han descubierto todavía que sus estudios, que en general suelen ser bastante tediosos y aburridos, podrían ser más llevaderos con el uso de la tecnología informática.

—En efecto, el *software* sin otro elemento que el lúdico, tiene por intención sólo la de divertir y pasar el rato. La mayoría de las veces es una descarga nerviosa y sólo exige una rapidez de reacción. De la misma manera que con otros juegos, llegan a aburrirse y los dejan de lado. Esta conducta que es habitual en los niños hace que también dejen de lado el ordenador, ya que sólo les servía como medio para cargar sus programas de juegos. Sólo una minoría manifiesta actitudes más creativas hacia el ordenador, y comienzan a aprender lenguajes de programación.

El uso de programas educativos, sin embargo, facilita el aprendizaje y la transmisión de conocimientos...

—Perdona la interrupción. Decías que comienzan a aprender lenguajes de programación. Sabemos que por lo general el BASIC es el lenguaje con el que empiezan. Pero aquí también se les cierran caminos. En primer lugar, porque su aprendizaje no está al alcance de todas las edades; y en segundo lugar, porque lo que quieren imitar, lo que les gustaría hacer, es decir, esos juegos que tanto les apasionan por su calidad gráfica, por su velocidad, y en definitiva por su complejidad técnica, requieren mucha profesionalidad; amén de estar escritos en su mayoría en lenguaje máquina; y la programación en lenguaje máquina es aún mucho más difícil.

Los resultados que pueden obtener son pobres respecto a sus expectativas, y un posible motivo de frustraciones.

La creación de programas que canalicen esta actitud creativa y que no les cierren puertas, podría ser el marco de un trabajo educacional. ¿No te parece?

—Efectivamente. La transmisión de conocimientos puede hacerse mediante programas de enseñanza clásicos en los que al chico se lo instruye a través de lecciones. Por lo que el uso no tiene para él, mayor interés que la novedad de te-

ner esta lección en una pantalla en lugar de un libro o del dictado de un profesor.

Hay otra manera en la que creemos más, destacando el uso interactivo y las posibilidades de simulación. Es decir, creando una serie de «mundos» que el chaval puede controlar y dirigir, y lo que es más importante experimentar tanto como quiera.

Esto nos obliga a elaborar y dar forma a todas estas ideas y conceptos que queremos trasmitir y con las que queremos que el niño trabaje.

El concepto de «mundo» está muy relacionado con el de microambientes del Logo. Es decir, debemos disponer un conjunto de posibilidades que se pueden manipular libremente y que están relacionadas con el conocimiento que queremos transmitir. Se me ocurre el ejemplo de *Números Locos*, donde los niños controlan un par de muñecos haciendo una carrera de saltos. Estos saltos responden a principios aritméticos. O como en *Teclas Divertidas*, en las que el niño tiene que evitar que desaparezcan las figuras que van cayendo por la pantalla escribiendo su nombres.

—Por lo general los chavales piensan que los programas educativos son aburridos...

—Nuestros juegos efectivamente son tan divertidos como cualquier otro con una diferencia fundamental: los personajes o elementos con los que trabajan les están formando a la vez que divirtiendo.

Podríamos preguntarnos hasta qué punto construir con un mecano, jugar al monopoli o resolver un puzzle no es apasionante o divertido.

Lo cierto es que nuestros programas están diseñados principalmente para divertir, y fundamentalmente para enseñar.

Por otra parte quienes normalmente compran los juegos a los niños son sus padres. Muchas veces no tienen en cuenta los beneficios potenciales que un ordenaodr le puede proporcionar a un niño. Es evidente que si sólo se le enseña a usarlo para jugar, cuando el juego pierde su interés, el ordenador, como máquina de juegos, también pierde interés.

—Tenemos entendido que en España se están comenzando a hacer cosas en este terreno.

—En nuestro país todavía existe la identificación confusa de uso educativo del ordenador igual a lecciones por ordenador.

Personalmente pienso que todavía falta un conocimiento más profundo de las posibilidades del ordenador. Este es fundamentalmente gráficos, sonido, e interactividad.

Si esto se tuviera en cuenta no encontraríamos títulos de progrmas como *Matemáticas 1-2*, y que nos recuerdan a novelas de entregas por fascículos.

En la elaboración de un programa educativo debe participar un equipo multidisciplinario de expertos que se encarguen cada uno de ellos del apartado en el que poseen experiencia. Es así como equipos como el de *Tom Snyder Corp., Joyce Hackasoon As., Interactive Picture Systems*, y otros muchos están trabajando y realizando los programas más avanzados en este campo.

Estos programas son de aventuras donde el personaje descubre la geografía o la historia de su país, realiza viajes a través de la mitología griega, o ha de construir un «pin-ball» teniendo en cuenta las reglas de los sistemas lógicos.

Evidentemente al niño no se le propone el estudio de la geografía, sino que se le introduce en el mundo de la fantasía que responde a la realidad.

Cualquiera de estos programas exige meses de experimentación y desarrollo. Siempre se intenta llegar a la versión más simple posible desde el punto de vista de su facilidad de uso. De esta manera no es de extrañar encontrar programas que aparentemente son simples, pero su pontecialidad educativa es inmensa. Una analogía podría ser el uso de la arcilla, que no exige casi ningún conocimiento previo, pero con la que se pueden hacer hasta obras de arte.

—Esto es fascinante. Especialistas de distintas ramas como pedagogos, grafistas e informáticos, trabajando para crear algo sencillo de usar pero potente en cuanto a sus resultados. Nos imaginamos que la valoración de este trabajo no suele ser muy buena. Somos muy afectos a dejarnos deslumbrar por aquello que parece complicado y difícil de hacer, porque difícil lo vemos en el resultado. Utilizar estos programas nos exige una cierta mentalización, y nos ofrece la posibilidad de usar el ordenador con otras perspectivas.

En nuestro país todavía existe la identificación confusa de uso educativo del ordenador igual a lecciones por ordenador.

—Pero actualmente, ¿qué estáis haciendo?

—Nuestro proyecto inmediato, después de haber sacado las dos primeras series de títulos de *Spinaker y Fisher Price,* es el lanzamiento de una nueva serie que convierte al usuario en protagonista de libros clásicos como la Familia Robinson, entre otros.

A medio plazo nos proponemos realizar programas de simulación para MSDOS y por supuesto seguimos trabajando en Logo por sus enormes posiblidades.

—Claro que para los maestros y profesores que trabajan en las escuelas, el poder hacerlo con estas perspectivas parecería difícil.

—Yo les recomendaría a los profesionales de la educación que están interesados en el uso de ordenadores en sus clases, que no se contentaran con realizar los típicos programas de aplicaciones matemáticas o de preguntas y respuestas, y que se lanzaran a experimentar con Logo y sus enormes posibilidades de simulación, que van mucho más allá de lo más conocido que son los gráficos de la tortuga.

A los profesionales de la educación les recomiendo que se lancen a experimentar con Logo y sus enormes posibilidades de simulación.

Logo permite crear programas a la medida de sus necesidades para que luego sus alumnos experimenten con ellos.

Nosotros trabajamos con diversos grupos de profesores y con escuelas. Resulta realmente sorprendente ver cómo al cabo de poco tiempo son capaces de diseñar procedimientos de simulación que en cualquier otro lenguaje les exigiría una formación técnica mucho más amplia y costosa.

—Respecto a la posibilidad de compartir experiencias en este terreno haría falta un marco adecuado para poder aprender y utilizar estos resultados en la escuela. ¿No crees que en este sentido estamos todavía un poco en pañales?

—Otro proyecto en el que estamos trabajando y que ponemos en marcha a partir de este mes el *Centro de Actividades-Logo y Taller de Informática Educativa.* Estará abierto a profesores, profesionales, y a todos aquellos interesados en conocer a fondo las posibilidades del ordenador en el medio educativo.

También los niños podrán trabajar en estos talleres de familiarización con la informática.

En este centro de carácter multidisciplinario se creará un entorno apropiado para que contemos con grupos de desarrollo de software educativo.

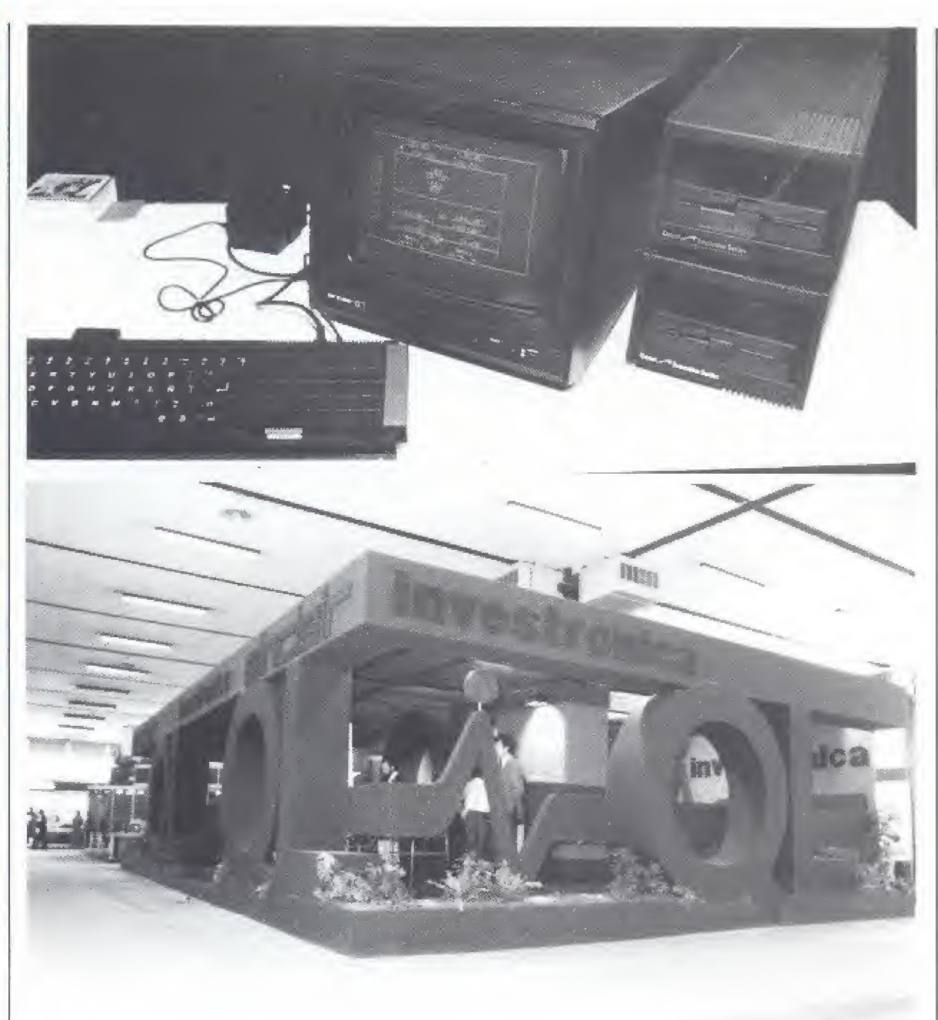
—Casi sin darnos cuenta ya han pasado más de dos horas de charla. Nos despedimos. Nos sentimos satisfechos de saber que aquí en España también queremos nuestros propios logros en este terreno. La sala de programadores ya no tiene luz. Por hoy ha terminado la actividad. Son casi las diez de la noche. Afuera está lloviendo.



Reportaje realizado por Miguel Figini en las instalaciones de Idealogic en Barcelona.



QL
MAGAZINE
Suplemento especial
junio 1985
Hoja electrónica con ABACUS
AJEDREZ TRIDIMENSIONAL



QL en el Informat

Como ya anticipábamos en el número anterior, la feria Informat supuso el lanzamiento de la versión española del QL. Primera versión de su «nueva generación de microordenadores en lengua local», como rezaba la publicidad.

Esta primicia española, se debe según palabras de **Charles Cotton,** responsable de la política internacional de Sinclair, a la importancia del castellano en el contexto internacional y al liderazgo de los ordenadores Sinclair en España.

En un stand llamativo cuyas paredes estaban formadas por la palabra QL, se presentaba el nuevo ordenador, junto con los discos de **Quest.** Pasada la sensación inicial, lo más llamativo resultaba ser la ausencia total de ordenadores Spectrum.

El precio definitivo es de 125.000 pesetas, incluidos los programas de **Psion,** notablemente mejorados. Entre las características principales destaca: mensaje del sistema, así como los avisos de error del operativo y de los cuatro programas de **Psion,** en castellano; acentuación de vocales; nuevos caracteres (! ¿ Ñ ñ ü ç).

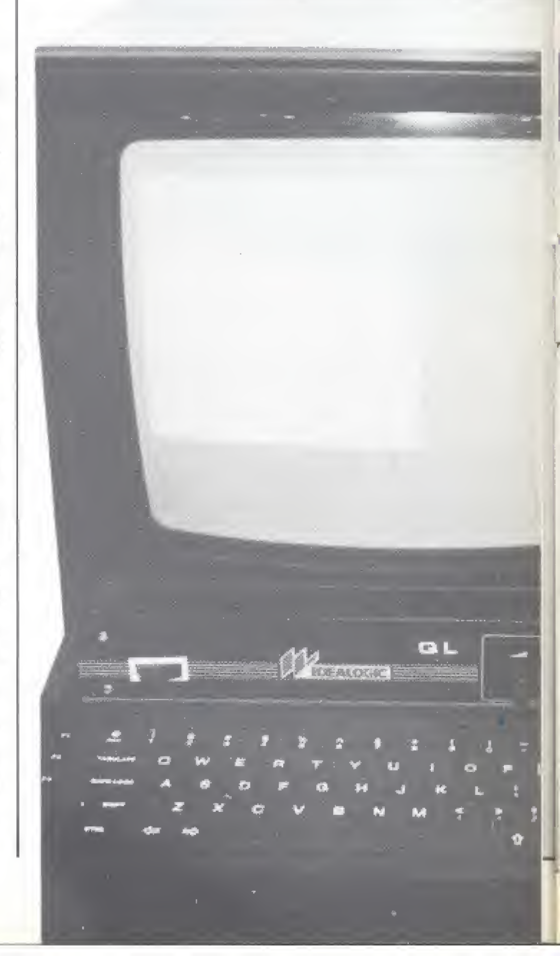
Con ocasión de esta feria, pudimos comprobar las mejoras sustanciales llevadas a cabo en los nuevos programas de **Psion.** El acceso continuo a *microdrive* prácticamente se ha eliminado, a excepción de las opciones de ayuda, lográndose así una rapidez considerable. Asimismo, se ha reducido el tamaño de los programas, permitiendo mayores ficheros para el usuario. En este sentido, las prestaciones de los programas quedan de la siguiente forma:

Software en marcha

Igualmente con motivo del Informat, **Investrónica** dio a conocer su decisión de apoyar fuertemente a este ordenador en el terreno del *software.* Los primeros programas que verán la luz serán el QL-MEDICINA, vademécum para archivar datos sobre medicamentos; QL-PORTICOS, cálculo, armado y medición de pórticos de edificación; QL-NOMINAS, hasta 500 nóminas parametrizables, permitiendo hasta 40 deventos y 10 deducciones.

Monitor «fuerte»

En principio, cualquier monitor puede servir para el QL. Sin embargo, por razones de nitidez debido a las 80 columnas,



QL-Aplicaciones

Los programas de **Psion** son verdaderamente interesantes, pero no lo son todos. Conscientes de ello, Sinclair anuncia la aparición de nuevos programas de aplicaciones: *Entrepreneur, Project Planner* y *Decision Maker.*

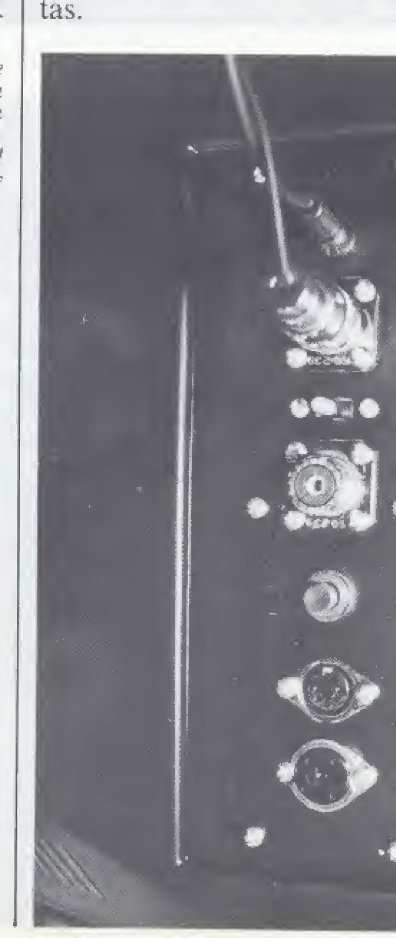
Entrepreneur para ayudar al ejecutivo en sus decisiones financieras y análisis de cash-flow; *Project Planner* para planificar distintas actividades y resolver el problema de los tiempos crítcos; y *Decision Maker* para conocer qué decisión tomar y a qué coste.

Estos tres programas han sido producidos por **Triptych Publishing Ltd.,** y se comercializarán en breve en el Reino Unido al precio de 39,95 libras (aproximadamente 8.000 pesetas) cada uno.

existen monitores «especiales» para este ordenador. Tal es el caso de este monitor de **Idealogic.**

Entre las características del monitor color de 14 pulgadas, especial-QL fabricado y comercializado por **Idealogic,** destaca precisamente la cualidad de resistir todo tipo de golpes.

De dimensiones 36 × 36 × 36, dispone de conexiones RGB y Video Pal. Precio: 75.000 pesetas.



Conectores laterales. De abajo a arriba: entrada sonido, video-pal, salida video, video pal, conmutador video pal RGB, entrada RGB y salida sonido.

Gráficos «a todo color»

La preocupación por el desarrollo de aplicaciones comerciales, ha dejado en segundo lugar las posibilidades gráficas de este ordenador. Para remediar este problema, la casa inglesa **Talent** la lanzado el GRAPHIQL.

El programa se suministra en dos *microdrives,* aunque el segundo tan sólo incluye tres pantallas de demostración. Distribuidor: Serma.

Juegos

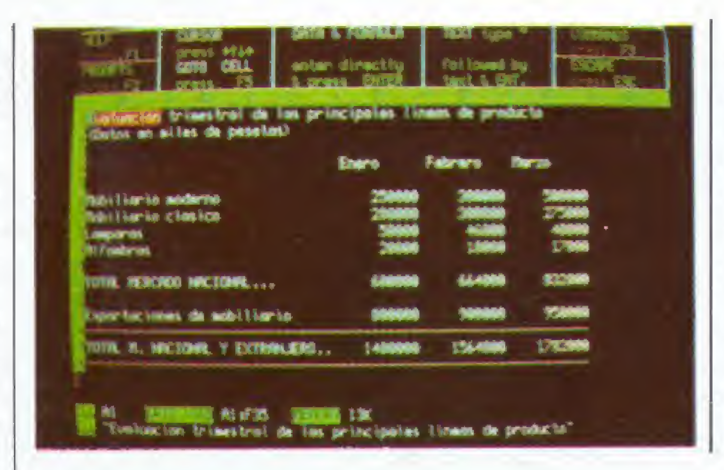
Apostando por el QL, **Serma** se convierte en la empresa pionera en material para este ordenador. Además del célebre ajedrez de **Psion**, acaba de incorporar a su catálogo de productos los primeros juegos aparecidos en Inglaterra para el QL, la mayor parte de ellos de aventuras: *The lost kingdom of Zkul* (aventuras: 7.500 pesetas), *West* (aventuras: 7.500 pesetas), *Graphiql* (utilidad: 12.000 pesetas) y *D-Day* («*war game*» o juego de guerra: 9.000 pesetas).





ABACUS, una hoja electrónica inteligente

Entre los cuatro programas incluidos con el QL no podía faltar una hoja electrónica. Y eso indica la creciente popularidad de este tipo de programas, a pesar del poco tiempo que han estado presentes estas aplicaciones en el mercado profesional.



En 1979 dos estudiantes de la **Harvard Business School** tenían que resolver problemas que incluían gran cantidad de datos interrelacionados a través de fórmulas. Se dieron cuenta de que cualquier pequeña variación en uno de los datos les exigía corregir toda la hoja en la que calculaban a base de lápiz, goma de borrar y calculadora. A menudo, el problema incluía cuestiones como ¿qué ocurre si el tipo de interés aumenta un 2 por 100? Eso les exigía recalcular toda la hoja, con nuevo riesgo de errores. Un trabajo muy pesado, que decidieron aliviar con ayuda de su microordenador, un Apple II. El resultado fue el famoso Visi Calc, del que se vendieron 400.000 ejemplares en tres años.

Su propósito era presentar en la pantalla tablas de números y operar con ellos cómodamente, con la capacidad de recalcular toda la tabla de una manera automática cada vez que se modifica una de sus entradas. Resultan muy útiles para profesionales que deban tomar decisiones en materia económica condicionadas por gran cantidad de factores. Alterando unas pocas entradas de la tabla y esperando unos instantes, se calcula un nuevo resultado acorde con las nuevas hipótesis.

La evolución de estos programas, auténticos *«Best Sellers»* del *software* Profesional, les ha llevado a incorporar funciones más y más complejas, y algunos teóricos afirman que cualquier problema que se pueda resolver mediante un programa de ordenador se puede resolver mediante una hoja elec-

«Cash Flow»: Un Ejemplo

Una de las aplicaciones clásicas de las hojas electrónicas de cálculo es la de resolver problemas de planificación financiera. Para mostrar su realización mediante ABACUS adaptamos uno de los ejemplos del manual. Como allí, antecedemos cada comando de una referencia de columna y fila. Esta referencia no se debe teclear; indica dónde debemos posicionar el cursor antes de introducir la fórmula. En algunas fórmulas, el programa nos pregunta en qué extensión de columnas o filas debe actuar el comando. En ese caso, indicamos entre corchetes la respuesta que se debe introducir. Pasemos a analizar nuestra inversión.

[C1] "CASH FLOW"
[C2] rept ("=", len (C1))

Estos dos comandos introducen el título y lo subrayan. Es siempre importante que nuestras hojas tengan un título que identifique su contenido.

[A4] row = month (col()–2) (columnas B a N)

trónica, aunque no siempre de una manera práctica.

ABACUS es la aportación de **Psion** en este campo y, aunque parece que ya no queda nada por inventar, sí incluye algunas mejoras sobre los programas de la generación anterior.

Cualquier hoja electrónica divide la memoria en celdas que componen una matriz rectangular, y la manera 'clásica' de referirnos a una celda es por una o dos letras (que especifican la columna en que nos movemos) y un número (que indica a qué altura de la columna queremos llegar). Los programas más complicados incluyen la posibilidad de darle nombre a columnas y filas, lo que permite una referencia más fácil de memorizar para nuestros datos, así como la inserción de texto en algunas de las celdas, muy útil para conseguir tablas legibles. El programa de **Psion** cumple esta condición, al poderse referenciar una celda mediante nombres, que hayan sido insertados en la fila y columna correspondientes. Mas aún, basta con indicarle al programa el comienzo del nombre, con las letras suficientes para identificar la fila o columna de manera única.



Cada una de las celdas puede contener tres tipos distintos de datos: números, texto o fórmulas. En caso de que la celda contenga una fórmula, el programa no la mostrará, sino que presenta en pantalla su resultado cada vez que la evalúe. Sin embargo, la fórmula aparece en la parte inferior de la pantalla, lo que nos permite distinguir la celda de aquéllas que contienen números. En caso de que la casilla contenga un texto, será éste el que se muestre, pudiendo extenderse a las celdas vecinas si están vacías.

Otra característica de cualquier hoja de cálculo (imprescindible para olvidadizos) es la posibilidad de añadir filas o columnas entre las que ya están ocupadas. ABACUS lo permite sin más restricción que la capaci-

[A5] row = rept ("=", width ()) (A a N)

Mediante estas dos líneas introducimos los encabezamientos con los meses del año, comenzando en diciembre (mes de la inversión inicial) y acabando en diciembre del año siguiente. Si nuestro programa es en castellano, los nombres de los comandos y meses serán en español, y las referencias a Ven.jan serán a Ven.ene.

[A6] "Ventas
[C6] 4000000
[D6] row = ven-.jan*1.02 (D a N)

Con estas dos instrucciones introducimos una expectativa de ventas para nuestro producto de 4 millones de pesetas para el mes de enero, y un incremento previsto de un 2 por 100 mensual a lo largo del año.

[A7] "Coste de ventas
[A7] row = ven.jan*0.5 + 750000 (C a N)
[A8] row = A5 (A a N)

El coste de las ventas es igual a la mitad de las ventas más un coste fijo de 750000 Ptas. La fórmula llena automáticamente todas las casillas del año con nuestras estimaciones iniciales. La fila A8 queda subrayada al hacerla igual a la fila 5.

[A9] "Bruto
[B9] row = ven - cos (C a N)

Con esta fórmula estimamos nuestro margen bruto, como la diferencia entre las cifras de ventas y las de coste para cada mes.

[A11] "Gastos
[A12] "Salarios
[C12] row = 700000 (C a N)
[A13] "publicidad
[C13] row = 100000 (C a N)
[A14] "alquiler
[C14] row = 200000 (C a N)
[A15] "Electricidad
[C15] row = 50000 (C a N)
[A16] "Inv./Mant.
[B16] 2500000
[C16] row = 90000 (C a N)
[A17] row = a5 (A a N)

Con estas líneas incluimos en nuestra hoja las previsiones de gastos no imputables a las ventas del producto. La inversión inicial, de dos millones y medio en nuestra estimación inicial, es la única cifra en la columna de diciembre del año anterior.



dad total, limitada a 15K en la versión 1 y ampliada a 23K en la versión 2. Las 15K de las primeras versiones permiten algo más de mil celdas ocupadas. Este número es relativamente pequeño, y resulta uno de los mayores inconvenientes del programa. Con la mejora en la versión 2 se ha dado un paso en la dirección correcta, aunque la ampliación de memoria resultará imprescindible si queremos trabajar con grandes tablas de números.

También podemos «abrir un hueco» en nuestro papel moviendo su contenido a otra

Las hojas electrónicas, auténticos Best Sellers del Software profesional

posición. ABACUS lo hace, bastando pulsar F3 luego C (de Copy) y especificar el rango de la hoja que será afectado por la copia (p. ej. a1:b3 o Cost.Feb:Vent-.Dic) y después el ángulo superior izquierdo de la nueva localización del bloque.

Para borrar basta usar el comando **Rubout** (R) y especificar un rango. Todas las casillas entre las dos dadas perderán su contenido. Otro comando imprescindible es **Echo** (E), que copia el contenido de una celda sobre un conjunto de casillas. Cuando se copian fórmulas, las referencias de casillas son siempre relativas, a menos que se especifique lo contrario precediendo la referencia con el signo del dolar ($).

El comando **Amend** (A) permite editar un número, fórmula o texto. Es importante tener en cuenta que las fórmulas duplicadas mediante **Echo** o **Copy** son copias de la misma fórmula maestra, por lo que la modificación de una de las celdas cambiará la fórmula a lo largo de toda la hoja.

Un punto a favor de ABACUS es el gran número de opciones de representación numérica de que dispone: se pueden representar números enteros, decima-

	A	B	C	D	E	F
1			CASH FLOW			
2			=========			
3						
4		December	January	February	March	April
5	==========	==========	==========	==========	==========	==========
6	Ventas		4000000	4080000	4161600	4244832
7	Coste de Ventas		2750000	2790000	2830800	2872416
8	==========	==========	==========	==========	==========	==========
9	Bruto		1250000	1290000	1330800	1372416
10						
11	Gastos					
12	Salarios		700000	700000	700000	700000
13	Publicidad		300000	200000	0	0
14	Alquiler		200000	200000	200000	200000
15	Electricidad		50000	50000	50000	50000
16	Inv. / Mant.	2500000	90000	90000	90000	90000
17	==========	==========	==========	==========	==========	==========
18	Total Gastos	2500000	1340000	1240000	1040000	1040000
19	==========	==========	==========	==========	==========	==========
20	Neto	-2500000	-90000	50000	290800	332416
21	==========	==========	==========	==========	==========	==========
22	Tasa Interna de Retorno			82		
23	Coste del dinero	18		Valor Actual Neto		3752808

Para mayor simplicidad, hemos supuesto constantes los gastos para todos los meses. Una vez construido el modelo, se puede variar el gasto en publicidad a lo largo de los meses, y ver cómo se alteran nuestras previsiones.

[A18] "Total Gastos

(12 a 16; B a N)

[A19] row = a5 (A a N)

[A20] "Neto

[B20] Neto = Brut - Tot. (B a N)

les, exponenciales o con notación monetaria o porcentual, con cualquier caracter como símbolo monetario. También se puede justificar texto o números a izquierda, derecha o centrado. Todas estas opciones se pueden aplicar a cualquier rango de celdas.

La gestión de archivos tiene los comandos necesarios para almacenar nuestro trabajo a salvo. Incluye también un comando para «Mezclar» dos tablas, sumando o restando de la hoja actual otra contenida en **microdrive.**

En cuanto a funciones de cálculo, ABACUS incorpora todas las funciones matemáticas previsiblemente necesarias, incluyendo trigonométricas y exponenciales, así como funciones que proporcionan promedios, sumatorios, tasas internas de retorno o el valor actual neto de una inversión, el día de la semana correspondiente a un mes dado, o indexan en una tabla un valor para interpolación. Incluso una función if, que proporciona valores dependientes de condiciones lógicas.

La opción de importación permite recibir datos de Easel o Archive

La exportación hacia Easel, Archive o Quill resulta útil para insertar las tablas en documentos, tratarlas como registros de una base de datos o realizar representaciones gráficas de los resultados. La opción de importación permite recibir datos de Easel o Archive. Así, la facilidad de tratamiento gráfico de Easel queda reforzada. También es útil poder calcular usando registros de una base de datos.

En resumen, ABACUS es una hoja de cálculo que combina características muy avanzadas con las virtudes de los programas «clásicos» en este campo. Los lectores que conozcan ya algún programa de este tipo disfrutarán con su uso. Quienes nunca hayan usado esta herramienta, no sabrán cómo han podido sobrevivir tantos años sin ella.

El total de gastos se construye sumando todas las partidas de gastos para cada columna. El margen neto se obtiene como diferencia entre el margen bruto y los gastos totales.

[A21] row = rept ("=", width ()) (A a N)

Si, una vez construidas las estimaciones mensuales de evolución de nuestro negocio, queremos extraer alguna conclusión significtiva, podemos utilizar dos funciones definidas por el programa: *irr (rango, período),* que nos da la tasa interna de retorno para el número de filas o columnas especificadas en el primer parámetro, considerando el período en meses que las separa. O *npv (rango, interés, período),* que nos da el valor actual neto, con un significado similar en sus parámetros.

[A22] "Tasa Interna de Retorno
[C22] irr (Neto, 1) (B a N)

El 1 especifica que se trata de cifras mensuales.

[A23] "Valor Actual Neto
[C23] askn ("Interés")
[D23]

(1-C23/1200)*npv

(Neto, C23, 1) (C a N)

La casilla C23 nos pregunta el tipo de interés estimado, y la casilla siguiente calcula el valor actual neto de la inversión. Esta cifra se debe comparar con la inversión inicial. Cada vez que se recalcule la hoja del programa nos pedirá una tasa de interés, que usará para calcular el valor neto.

Finalizamos mejorando el formato de nuestro ejemplo:

F3 Grid, Width, 15 (A a A)

F3 Justify, Cells, Text, Right, A4:N4

F3 Justify, Cells, Text, Right, A12:A16

F3 Units, Cells, Integer, A1:N23

Si cambiamos, a medida que evoluciona nuestro negocio, alguna de las cifras de ventas, las previsiones de los meses siguientes se construirán a partir de esa cifra, ya que la fórmula siempre utiliza el mes anterior y le añade el 2 por 100.

Ajedrez con QL

Sorprendentemente, uno de los programas que han roto el monopolio de las aplicaciones «profesionales» para el QL es el Ajedrez de Psion. Un programa que empató con tres máquinas de ajedrez en el primer puesto del Campeonato del Mundo 1984, celebrado en Glasgow.



La presentación tridimensional del tablero ya nos indica que el programa se sale de lo corriente. Muy estudiada, su realización requirió la escritura previa de un programa de gráficos que permitiera el trabajo directo con el ordenador. Y es que, como dice Ann Hughes, la diseñadora gráfica del programa, «al principio lo hacía todo sobre papel gráfico, pero la pantalla tiende a aplanar los dibujos. Ahora que estoy acostumbrada hago casi todo mi trabajo directamente en pantalla». Y lo hace muy bien, como prueban las fotos que acompañan este artículo.

Se puede elegir entre la representación tridimensional, con

Ganó el QL

Efectivamente, le pusimos a prueba jugando contra uno de los programas más conocidos del mercado: el Sargon III, corriendo en uno de los ordenadores más atractivos en cuanto a diseño, el Macintosh.

Elejimos el nivel 3 en ambos programas, lo que supone jugar a quince segundos por movimiento para ambos programas. Jugamos dos partidas. En la primera salió el QL y necesitó 64 movimientos para derrotarle. En la segunda, las blancas las llevaba el Macintosh, y esta vez le costó 94 jugadas. Reproducimos esta segunda y encarnizada lucha.

Una advertencia final: recuerde que los programas de ajedrez tienen más moral que el Alcoyano y nunca dan por perdida una partida, aunque usted se hubiese retirado antes.

QL CHESS BY PSION

1	E2-E4	
1	. . .	D7-D6
2	D2-D4	
2	. . .	G8-F6
3	B1-C3	
3	. . .	G7-G6
4	G1-F3	
4	. . .	F8-G7
5	F1-E2	
5	. . .	E8-G8
6	E1-G1	
6	. . .	C7-C6
7	C1-F4	
7	. . .	F6-G4
8	E2-C4	
8	. . .	B7-B5
9	C4-B3	
9	. . .	E7-E5
10	F4-G5	
10	. . .	D8-C7
11	D1-D2	
11	. . .	H7-H6
12	G5-H4	
12	. . .	B8-A6
13	D4xE5	
13	. . .	G4xE5
14	F3xE5	
14	. . .	D6xE5
15	A1-D1	
15	. . .	A6-C5
16	H4-D8	
16	. . .	C7-B7
17	F2-F3	
17	. . .	C5xB3
18	A2xB3	
18	. . .	C8-E6
19	D8-H4	
19	. . .	A7-A5
20	D2-D6	
20	. . .	B5-B4
21	C3-A4	
21	. . .	B7-B5
22	A4-C5	
22	. . .	F8-E8
23	C5xE6	
23	. . .	E8xE6
24	D6-D2	
24	. . .	E6-E8
25	H4-F2	
25	. . .	E8-F8
26	F2-E3	
26	. . .	G8-H7
27	H2-H3	
27	. . .	F8-H8
28	F1-F2	
28	. . .	F7-F5
29	E4xF5	
29	. . .	G6xF5
30	D2-D7	
30	. . .	F5-F4
31	E3-A7	
31	. . .	H8-F8
32	F2-D2	
32	. . .	F8-G8
33	A7-F2	
33	. . .	A5-A4
34	B3xA4	
34	. . .	B5xA4
35	B2-B3	
35	. . .	A4-B5
36	D2-D6	
36	. . .	G8-C8
37	D7-F5	
37	. . .	H7-H8
38	F5-G6	
38	. . .	C6-C5
39	D6-D7	
39	. . .	C8-G8
40	D1-D5	
40	. . .	A8-A1
41	D5-D1	
41	. . .	A1xD1
42	D7xD1	
42	. . .	G8-A8
43	G6-E4	
43	. . .	A8-C8
44	D1-D6	



sólo el tablero en pantalla, o bien el formato clásico, con las últimas jugadas hechas, el reloj y una serie de informaciones sobre las opciones en vigor, que incluyen la selección entre catorce niveles. Estos se desdoblan en veintiocho por la opción *Easy* (fácil), que no deja pensar al ordenador mientras estudiamos nuestra jugada. Para uso en análisis de problemas, dispone de otros ocho niveles de estudio de mate (de una a ocho jugadas).

También permite la utilización de impresora para reflejar el resultado de la partida, imprimiendo las jugadas según se van produciendo o una representación del tablero en un momento dado de la partida. El almacenamiento de partidas en *microdrive* permite continuar jugadas largas o, mediante la opción *Replay,* analizar alternativas de juego en partidas jugadas anteriormente.

Muy útil resulta también la posibilidad de volver atrás el número de jugadas deseado. Menos corriente es encontrar la opción *Next-Best,* que hace al programa volver atrás una jugada y seleccionar el siguiente mejor movimiento. Si queremos encontrar alternativas de juego en momentos clave, su uso resultará muy cómodo, especialmente al combinarlo con la opción *Análisis,* que enseña en una línea en la parte inferior de la pantalla ls dos o tres jugadas siguientes y una evaluación de la posición. Las opciones *Hint* (Pista) y *Best* (Mejor) nos ayudan a realizar nuestra jugada y nos permiten saber qué está «tramando» nuestro oponente en la próxima jugada.

El programa viene dotado de casi todas las opciones clásicas para este tipo de programas. Pero lo verdaderamente sorprendente es la calidad de su juego. Incluso al nivel 3 (el elegido por defecto), el programa muestra una gran solidez en las aperturas y un desarrollo correcto de los finales, que es quizá la parte del juego donde más dejan que desear los programas de ajedrez por regla general.

Psion ha prometido realizar una versión del mismo programa para ordenadores mayores, como el Macintosh y el IBM PC. Este potente programa corriendo sobre un Macintosh (bastante más rápido que el QL) puede darle un susto a más de un aficionado de nivel alto.

Con el Spectrum el problema es demostrar que sirve para algo mas que para jugar. Con el QL sin embargo sucede lo contrario, y este buen programa es sólo el principio.

Distribuidor en España: Serma
Monitor utilizado: Novex NC-1414-CL

44	. . .	B5-A5
45	E4-G6	
45	. . .	A5-A7
46	D6-D5	
46	. . .	A7-A1
47	G1-H2	
47	. . .	A1-C3
48	D5-D7	
48	. . .	C8-G8
49	D7-C7	
49	. . .	C3-A1
50	F2xC5	
50	. . .	A1-C3
51	C5-B6	
51	. . .	C3-E1
52	C7-D7	
52	. . .	E1-F1
53	B6-C5	
53	. . .	F1-B5
54	D7-C7	
54	. . .	B5-A5
55	G6-C6	
55	. . .	G8-B8
56	C6-D7	
56	. . .	B8-G8
57	D7-E7	
57	. . .	A5-A1
58	C5xB4	
58	. . .	A1-D1
59	E7-D7	
59	. . .	D1xD7
60	C7xD7	
60	. . .	G8-B8
61	B4-D6	
61	. . .	B8-B6
62	G2-G4	
62	. . .	F4xG3
63	H2xG3	
63	. . .	H8-G8
64	G3-G4	
64	. . .	G7-H8
65	G4-F5	
65	. . .	B6-A6
66	D6xE5	
66	. . .	H8xE5
67	F5xE5	
67	. . .	A6-B6
68	F3-F4	
68	. . .	B6-B5
69	E5-E6	
69	. . .	B5-B4
70	F4-F5	
70	. . .	B4-H4
71	D7-D3	
71	. . .	H4-E4
72	E6-F6	
72	. . .	E4-B4
73	F6-G6	
73	. . .	B4-B6
74	F5-F6	
74	. . .	B6-B8
75	C2-C4	
75	. . .	B8-F8
76	D3-F3	
76	. . .	F8-F7
77	G6xH6	
77	. . .	F7-C7
78	H6-G6	
78	. . .	C7-G7
79	G6-F5	
79	. . .	G7-C7
80	F5-E6	
80	. . .	G8-F8
81	E6-D6	
81	. . .	C7-A7
82	C4-C5	
82	. . .	A7-A2
83	C5-C6	
83	. . .	A2-D2
84	D6-C5	
84	. . .	F8-F7
85	C6-C7	
85	. . .	D2-E2
86	C5-C6	
86	. . .	E2-C2
87	C6-B7	
87	. . .	C2-C5
88	C7-C8/Q	
88	. . .	C5xC8
89	B7xC8	
89	. . .	F7-E6
90	F6-F7	
90	. . .	E6-E5
91	F7-F8/Q	
91	. . .	E5-D5
92	C8-D7	
92	. . .	D5-D4
93	F8-E7	
93	. . .	D4-D5
94	F3-D3	





EXISTIENDO tanta variedad de modelos y precios en el mercado, en cuanto a *joystick* se refiere, no hay excusa para seguir utilizando las «sufridas» teclas del Spectrum. Si aún así el precio le parece excesivo, le proponemos hacerse uno «a su medida», con el que poder manejar con gran comodidad y utilidad, sus programas favoritos, seleccionar un menú en un programa de gestión de forma rápida y sencilla, o utilizar programas para dibujar gráficos en pantalla. Sólo tiene un inconveniente: ¡sus dimensiones son superiores a las del propio Spectrum!

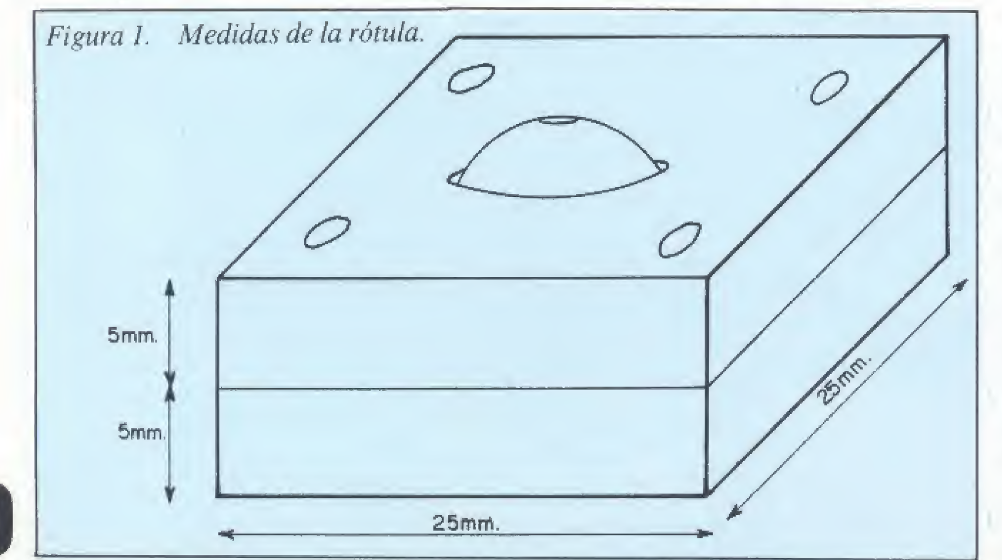
La palanca de mando aquí descrita ha sido diseñada para estar albergada en una caja de dimensiones anchas, en la cual se han atornillado dos pequeñas planchas de plomo y pegado a su base dos tiras de goma para así conseguir una adherencia más que suficiente sobre cualquier tipo de «terreno». Así se evita la molestia de tener que sujetar su base con la otra mano o, cuando menos, se tiene la sensación de manejar algo que se parece más a un accesorio útil que a un juguete.

El funcionamiento de la palanca es muy sencillo. Consiste únicamente en un eje que, al bascular, presiona los pulsadores sobre la propia tapa superior de la caja, permitiendo la pulsación simultánea de cada interruptor y de otro que esté situado perpendicularmente a éste, en función de la posición de la palanca.

El *joystick* consta únicamente de 5 interruptores (pulsadores) que son los que corresponden a las posiciones arriba, abajo, derecha, izquierda y disparo, y de una rótula que permite el movimiento libre de la palanca en todas las posiciones.

Los materiales utilizados para la realización de esta palanca han sido: madera de contrachapado de

Figura 1. Medidas de la rótula.





piezas, se pegará una fina placa de plástico, o cualquier material liso, sobre la parte interior de la tapa superior para conseguir que los alambres de los pulsadores deslicen sin dificultad. Por último, lijaremos todo el conjunto para obtener un buen acabado. Un par de manos de pintura finalizarán el montaje de la caja.

palanca puede efectuar cualquier movimiento en cualquier dirección. Está construida con dos placas de plexiglás de cinco milímetros que han sido lijadas hasta obtener una cavidad en forma de franja semiesférica en cada una de ellas (Fig. 1). De esta forma, al unirse, pueden albergar una esfera (en nuestro caso ha sido utilizada una canica de plástico) que puede moverse o girar con toda libertad de movimiento según los tres ejes cartesianos, pero sin poderse salir de las dos placas que la sujetan.

Montaje

Construya su Joystick

tres milímetros, un poco de plástico rígido, tipo plexiglás, un tubo de aluminio, cinco pulsadores, un conector, el correspondiente cable de conexión y algún que otro elemento más, que citaremos durante la descripción del montaje.

Construcción de la caja

Para comenzar, cortaremos todas las piezas de madera de contrachapado que constituyen el soporte de la palanca. En total consta de 18 piezas, incluida la base, que es la única pieza que se atornillará a todo el conjunto una vez finalizada la construcción. Las medidas (en centímetros) podremos obtenerlas directamente de la figura 5. Así pues todas las piezas van pegadas excepto la base, que se atornillará. Para ello, pegaremos unos taquitos de madera en el interior de las tapas laterales de la caja. Una vez que se hayan ensamblado todas las

Construcción de la rótula

Esta pieza es el «corazón» del *joystick*, ya que, gracias a ella, la





Construcción y montaje del eje

Una vez finalizada la rótula y fijada ésta a la carcasa, nos guiaremos de la figura 2, en la cual se aprecia un corte esquemático de todo el eje.

En primer lugar atornillaremos una placa de plexiglás de tres milímetros, cuyas dimensiones pueden extraerse de las figuras 3 y 4 en la base del tornillo, que servirá para soportar los pulsadores. Seguidamente, pasaremos el tornillo con una tuerca atornillada, sobre el taladro efectuado en la esfera del mismo diámetro que el del tornillo. Fijaremos el tornillo a la esfera con otra tuerca y, tras poner cuatro o cinco arandelas, atornillaremos la palanca de mando, que es un tubo de aluminio de ocho centímetros de longitud y dos centímetros de diámetro. A éste se la ha pegado una pequeña tapa circular de cualquier materia para poder fijar la tuerca en esta última, sobre la cual se atornillará el eje. El tubo tendrá un pequeño orificio para poder efectuar la salida del cable que va conectado al pulsador de disparo. Este pulsador irá pegado en la parte superior de la palanca con cual-

Figura 2. Montaje del eje.

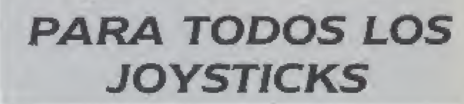
A) PULSADORES
B) ROTULA
C) PLACA SOBRE LA QUE SE DESLIZAN LOS ALAMBRES DE LOS PULSADORES

7 cm.

"SERMA" PRESENTA

COMO SACARLE TODO EL JUGO A SU QL

PARA TODAS LAS IMPRESORAS









CABLE CENTRONICS CON SOFTWARE INCLUIDO EN EL INTERFACE

CABLE RS 232 SUPER ECONOMICO SUPER CALIDAD

ADAPTADOR PARA JOYSTICK

I M P R E S C I N D I B L E S







LO NUNCA VISTO EN TRES DIMENSIONES 26 NIVELES ALTA DEFINICION

2 MICRODRIVES PARA CREAR DIBUJOS Y GRAFICOS

2 MICRODRIVES PARA EL MEJOR JUEGO DE GUERRA CON GRAFICOS COLOSALES

SERMA. VELAZQUEZ, 46 - 28001 Madrid. 431 39 11 - 431 39 74

TAMBIEN A LA VENTA EN TODAS LAS TIENDAS DE MICROINFORMATICA

PIDALOS POR TELEFONO O ENVIANDO ESTE CUPON

CANT.	CONCEPTO	PRECIO	CANT.	CONCEPTO	PRECIO
	C. CENTRONICS	9.995 pts.		AJEDREZ QL	6.000 pts.
	C. RS 232	5.250 pts.		GRAPHIC QL	12.000 pts.
	ADAPTADOR JOYSTICK	2.500 pts.		D.DAY	6.700 pts.

NOMBRE TEL CIUDAD

"SERMA"
Velázquez, 46
28001 Madrid

quier tipo de resina de dos componentes, por ejemplo. El otro extremo del cable pasará por otro pequeño orificio en la tapa superior.

Para conseguir que la palanca vuelva a la posición central cada vez que la soltamos, se utilizarán cuatro muelles que irán fijados desde cada lado de la placa, sobre la que se fijan los interruptores, a la tapa lateral de la caja enfrentada correspondiente (Fig. 3).

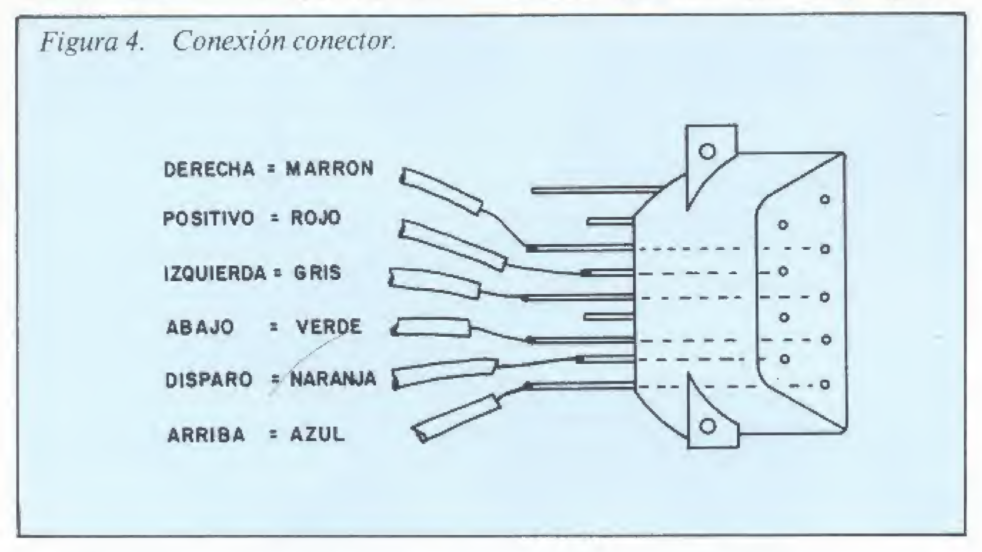


Figura 4. Conexión conector.

Montaje final

Por último, sólo nos queda soldar los interruptores a los cables que van al conector del *joystick,* según muestra también la figura 3.

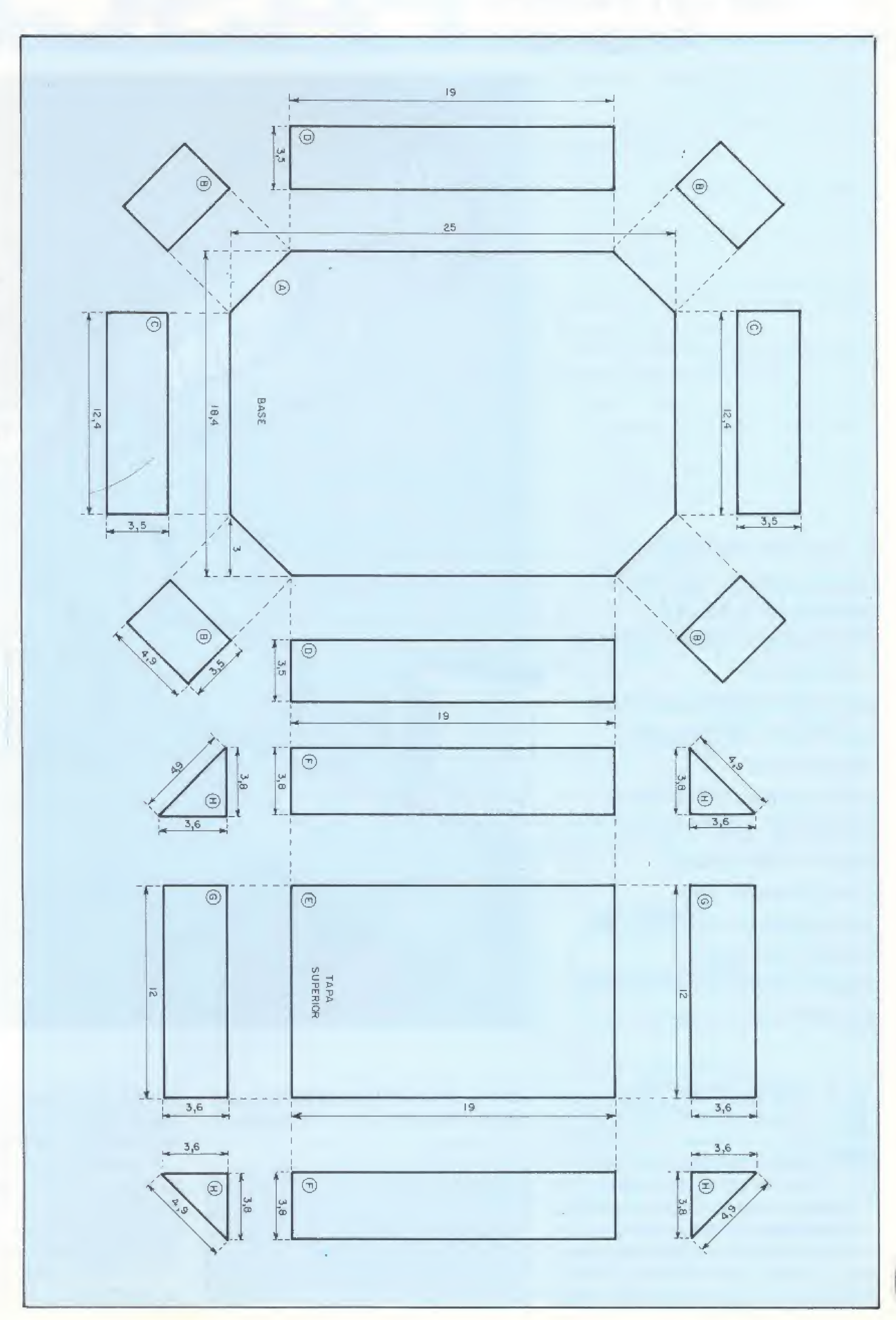
En nuestro prototipo se han atornillado en el interior dos pequeñas planchas de plomo en la base para evitar, en lo posible, el deslizamiento. Esto, naturalmente, es optativo.

Tal y como está conectado el *joystick* es absolutamente compatible con cualquiera de los que se pueden comprar en el mercado.

Alberto Piedra

Figura 3. Montaje final del joystick.

Figura 5. La caja consta de un total de 18 piezas, incluida la base, única pieza que se atornilla a todo el conjunto, a diferencia de las demás que van pegadas. En esta figura se pueden ver las dimensiones en centímetros de las distintas piezas que componen el montaje.



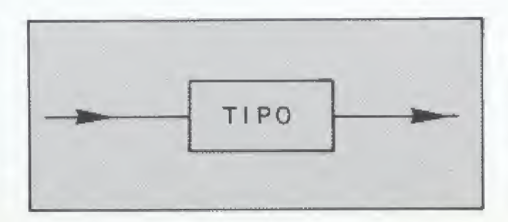
y 6

Descubrimiento de un nuevo lenguaje: PASCAL

Y con las matrices, finalizamos esta serie sobre el PASCAL. Esperamos que os haya sido útil el descubrimiento de este ya «viejo» lenguaje. Ahora que «dominamos» este lenguaje, estamos preparados para comprender los programas escritos en PASCAL que aparecerán en próximos números.

En las siguientes páginas que componen este artículo, vamos a explicar otro tipo de almacenamiento de datos. Son las matrices o tablas.

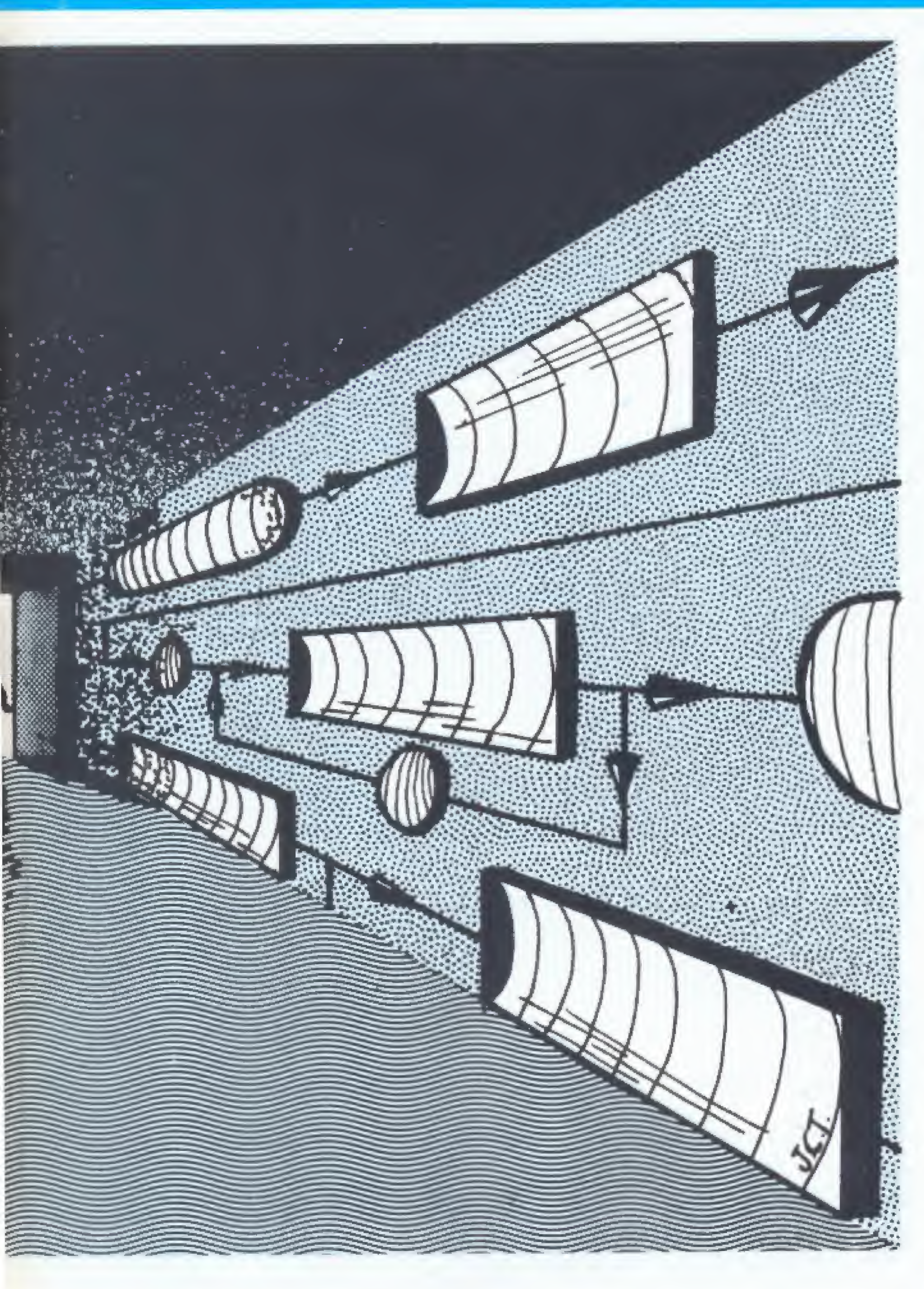
Cuando manejamos variables, normalmente se nos puede presentar un problema. Solamente podemos acceder a un valor de la misma durante la ejecución de un programa. Sin embargo, es posible que deseemos tener una colección de



valores en memoria para una misma variable. Mediante la estructura de tablas lo podemos conseguir.

Una definición formal de matriz sería un conjunto ordenado de elementos tal que un conjunto de enteros nos definen la posición de los elementos dentro de la misma.

Por tanto, como vemos, tenemos la posibilidad de acceder directa-



mente a un elemento conociendo la posición que éste ocupa dentro de nuestra tabla. Pero pasemos directamente a las matrices en Pascal. Las tablas son llamadas ARRAY, y su principal característica es que todos sus componentes han de ser del mismo tipo, al que llamaremos tipo base.

En general y teóricamente podremos tener matrices de «n» dimensiones. Entrando ya en la parte de declaración, es necesario indicar el nombre de la tabla, los extremos inferior y superior que nos indican la dimensión, y el tipo de los elementos que la componen.

Así en general declararemos una tabla de la siguiente forma:

```
VAR
nombre : ARRAY [i1..in, j1..jn]
         OF Tipo
```

Ejemplos de declaración de tablas podrían ser:

```
VAR
meses: ARRAY [1..12] OF
CHAR;
tabla: ARRAY [1..3, 1..3] OF
INTEGER;
```

Como ya sabemos mediante declaraciones TYPE también podemos designar tablas.

Otra forma de expresar los ejemplos anteriores podría ser:

```
TYPE
   Tmeses: ARRAY [1..12] OF
CHAR;
   Ttabla: ARRAY [1..3, 1..3]
OF INTEGER;
VAR
   meses: Tmeses;
   tabla: Ttabla;
```

Por otra parte también es posible dar un nombre a un intervalo (dimensión) mediante declaraciones TYPE.

```
TYPE
   dimension: 1..12
   Tmeses: ARRAY [dimension] OF CHAR;
VAR
   meses: Tmeses;
```

La figura 1 ilustra el diagrama sintáctico de una variable de tipo ARRAY, SUBINDICE y ELEMENTO.

```
TIPO ARRAY:
ARRAY
[ , ]
TIPO SUBINDICE
OF
TIPO ELEMENTO
OF
TIPO SUBINDICE:
      TIPO ORDINAL
TIPO ELEMENTO:
TIPO
```

Veamos ahora un ejemplo para el manejo simple de una tabla. Escribiremos un programa que primero llena una tabla con una palabra, para luego escribirla al revés.

Utilizaremos instrucciones FOR para llenar la tabla y escribirla

siendo uno de ellos descendente (ver programa 1).

Como vemos es un programa muy sencillo. Vamos a complicarlo un poco, generalizándolo para «n» letras. Supongamos que en nuestro programa ese «n» es 80 (programa 2).

Como se puede observar, el programa es prácticamente similar al anterior. Solamente utilizamos la variable «último» para detectar dentro de la tabla cuál es el último caracter de la línea. Para ello inicializamos dicha variable a 81 y vamos decrementándola hasta encontrar un caracter que sea distinto de BLANCO. Una vez encontrado éste, mediante el siguiente FOR descendente, escribimos la matriz invertida utilizando nuestro índice.

Hemos de insistir en que cualquier tipo ordinal puede elegirse como tipo subíndice, ya sea char, boolean, integer, real, tipo enumerado o tipo subrango. De esta forma las siguientes declaraciones de tablas serían sintácticamente correctas:

```
VAR
  palabra: ARRAY [CHAR] OF
INTEGER;
siglo: ARRAY [1800..1899] OF
REAL;
  ...........................
```

El tipo subíndice por tanto determina el número de elementos de la tabla.

Vamos a ver ahora que mediante una instrucción podemos manipular el conjunto formado por una tabla. Lo podremos conseguir mediante la asignación entre tablas. Para poder hacer una asignación entre dos tablas, es necesario que sean de características semejantes. Supongamos el siguiente ejemplo: Sean las matrices tabla 1 y tabla 2 definidas de la siguiente forma:

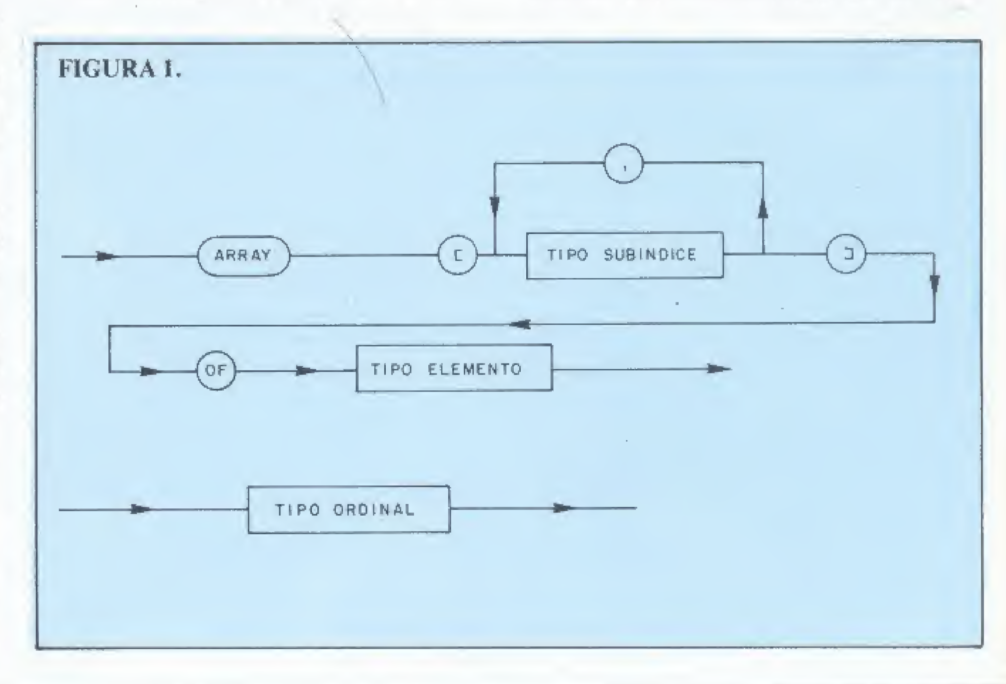
```
  ...........................
  ...........................
TYPE
  TIPO: ARRAY [1..12] OF
CHAR;
VAR
  TABLA1, TABLA2: TIPO;

  ...........................
  ...........................
```

En estas dos tablas podremos utilizar una sentencia del tipo:

```
TABLA1:=TABLA2;
```

de tal forma que a cada elemento de la TABLA1 le asignamos el elemento correspondiente de la TA-



BLA2, quedando las dos tablas iguales después de la asignación. Hemos manipulado el conjunto de elementos de las tablas al mismo tiempo.

Hasta ahora hemos estado utilizando ARRAYS de una sola dimensión, también llamados vectores. Pero también podremos definirnos ARRAYS multidimensionales, dependiendo la máxima dimensión del compilador que utilicemos.

Una definición de array multidimensional será:

VAR
  tabla: ARRAY [i..j] OF ARRAY [k..l] OF TIPO;

La forma abreviada de expresar la anterior declaración sería:

VAR
  tabla: ARRAY [i..j, k..l] OF TIPO;

Para comprender un poco mejor las matrices multidimensionales, vamos a realizar un sencillo programa que sume dos tablas de 2 dimensiones, guardando el resultado de dicha suma en otra tabla.

Haremos la lectura, escritura y suma en bloques diferentes del programa para que se vea mejor la estructura. Por supuesto este programa se podría simplificar mucho más (ver programa 3).

Por último, comentar que en algunos compiladores es necesario inicializar las tablas a ceros o blancos dependiendo del tipo, por si al rellenar las matrices no completamos todos sus elementos, y los vacíos contengan cosas que no nos interesan y que ya estaban con anterioridad en memoria.

José Ramón Herreros

PROGRAMA 1

```
PROGRAM verres;
VAR
   letras: ARRAY [1..10] OF CHAR;
   indice: INTEGER;

BEGIN
     READLN;
     FOR indice:=1 TO 10 DO
        READ(letras[indice]);
     FOR indice:=10 DOWNTO 1 DO
        WRITE(letras[indice])
END.
```

PROGRAMA 2

```
PROGRAM verres;
CONST
     NCT=80; (* NUMERO DE CARACTERES TOTALES *)
     BLANCO=' ';

VAR
   letras: ARRAY [1..NCT] OF CHAR;
   indice:INTEGER;
   ultimo:INTEGER;

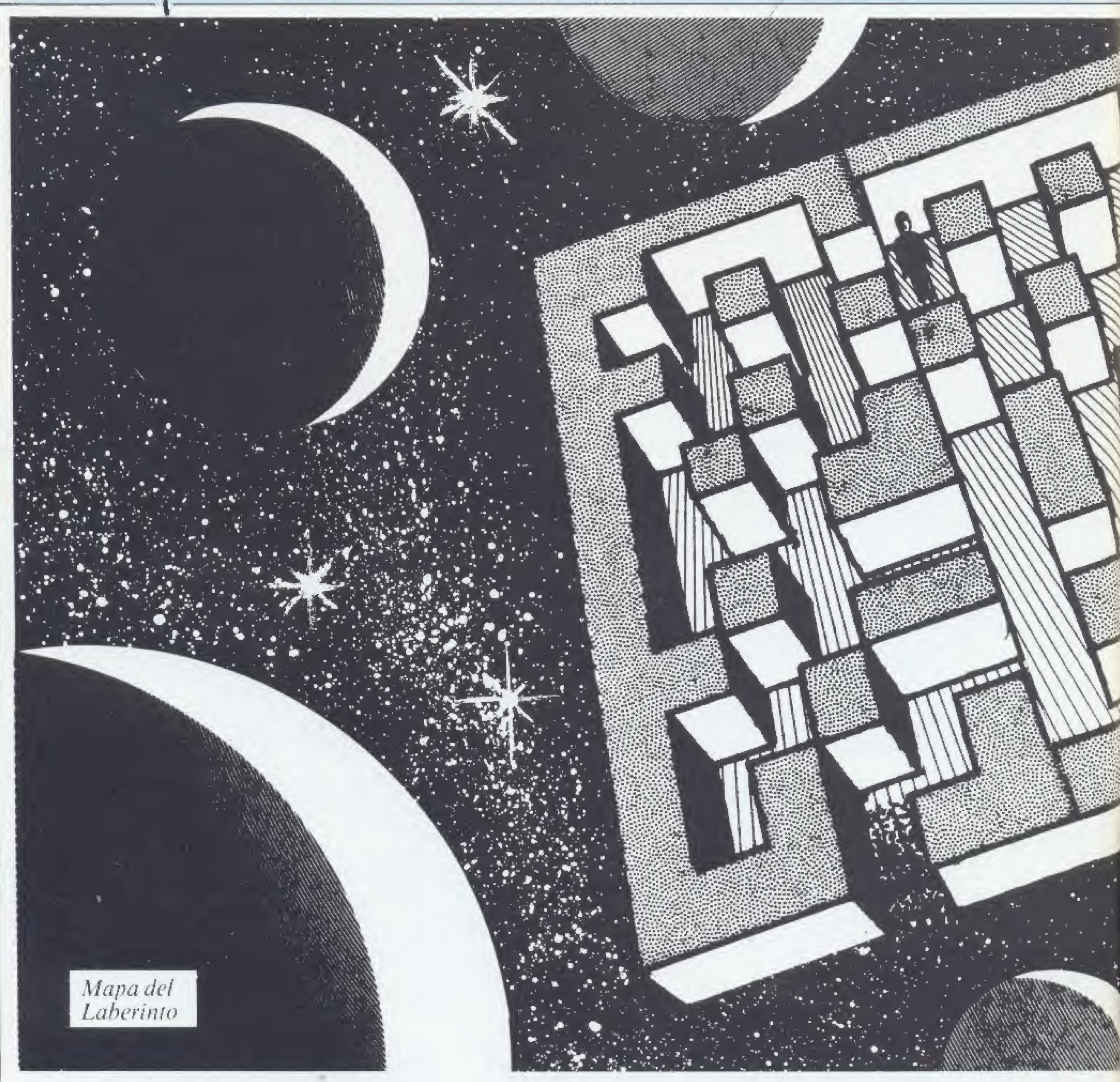
BEGIN
     READLN;
     FOR indice:=1 TO NCT DO
        READ(letras[indice])
     ultimo:=81;
     REPEAT
          ultimo:=ultimo-1
     UNTIL letras[ultimo]<>BLANCO;
     FOR indice:=ultimo DOWNTO 1 DO
        WRITE(letras[indice])
END.
```

PROGRAMA 3

```
PROGRAM SUMAR;

VAR
   A,B,SUMA: ARRAY [1..5,1..5] OF REAL;
   I,J:INTEGER;

BEGIN (*PROGRAMA PRINCIPAL*)
        FOR I:=1 TO 5 DO
        BEGIN
            FOR J:=1 TO 5 DO
            BEGIN
                READ(A[I,J]);
                READ(B[I,J])
            END
        END;
        FOR I:=1 TO 5 DO
        BEGIN
            FOR J:=1 TO 5 DO
                SUMA[I,J]=A[I,J]+B[I,J]
        END;
        FOR I:=1 TO 5 DO
        BEGIN
            FOR J:=1 TO 5 DO
                WRITE(SUMA[I,J])
        END
END.(*PROGRAMA*)
```



Mapa del Laberinto

Laberinto

Este programa tiene un origen muy peculiar: nació en un Commodore en la Escuela de Ingenieros de Zaragoza. Su autor, Tomás Plou, abandonó sus estudios de Ciencias Empresariales porque «para trabajar hay que tener experiencia y con un título no es suficiente». Ahora se prepara las oposiciones a informática de la Seguridad Social: «Si hay que trabajar, mejor hacerlo en lo que a uno le gusta, y la informática es apasionante, sobre todo estos temas de la quinta generación, la inteligencia artificial».

En el cursillo de BASIC de la Escuela de Ingenieros nació, como decíamos, este programa. Primero para el Commodore y posteriormente trasladado al Spectrum. «El mayor problema fue la matriz de





```
  30 CLS
  31 LET M=0: LET N=0
  40 PAPER 5: BORDER 6: INK 2
  42 PRINT AT 6,3;"MUEVES CON LA
S TECLAS 5-8"
  43 PRINT AT 8,3;"0= AYUDA"
  44 PAUSE 200
  45 CLS
  50 DIM L(12,12)
  55 FOR I=1 TO 12
  60 FOR J=1 TO 12
  65 READ L(I,J)
  69 NEXT J
  70 NEXT I
  71 DATA 1,1,1,1,1,1,1,1,2,1,1,
1
  72 DATA 1,0,0,0,0,0,1,0,0,1,0,
1
  73 DATA 1,0,1,0,1,0,0,0,1,0,0,
1
  74 DATA 1,0,1,0,0,0,1,1,0,0,1,
1
  75 DATA 1,0,1,0,1,0,0,0,0,1,0,
1
  76 DATA 1,0,1,0,1,0,1,1,0,0,0,
1
  77 DATA 1,0,0,0,0,0,1,0,0,1,0,
1
  78 DATA 1,1,0,1,0,1,0,0,1,0,0,
1
  79 DATA 1,0,0,0,0,0,1,0,0,0,1,
1
  80 DATA 1,0,1,0,1,0,0,0,1,0,0,
1
  81 DATA 1,0,0,0,0,0,1,0,0,0,1,
1
  82 DATA 1,1,1,1,1,1,1,1,1,1,1,
1
```

datos del laberinto. Para explorar la matriz tuve que crear un algoritmo que se puede ver en la línea 540. Dependiendo de la orientación, había que acudir a las mismas subrutinas de dibujo del laberinto. Aún así no tuve memoria suficiente y tuve que reducir algunas sentencias.»

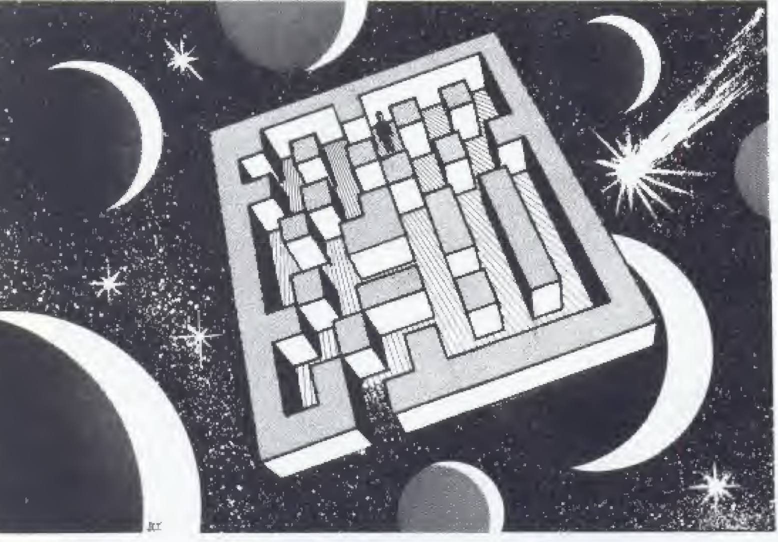
Autor: Tomás Plou *16K*



Programas

(46)

```
 104 LET I=INT (6*RND)+6
 105 LET J=INT (10*RND)+2
 110 IF L(I,J)=1 THEN  GO TO 104
 120 LET S=INT (4*RND)+1
 130 GO TO 156
 150 IF INKEY$="" THEN  GO TO 15
0
 151 LET Q$=INKEY$: IF Q$<>"5" A
ND Q$<>"6" AND Q$<>"7" AND Q$<>"
8" AND Q$<>"0" THEN  GO TO 150
 152 LET Q=VAL INKEY$: CLS
 154 IF Q<>5 THEN  GO TO 158
 155 LET S=S-1: IF S<1 THEN  LET
 S=S+4
 156 GO TO 170
```

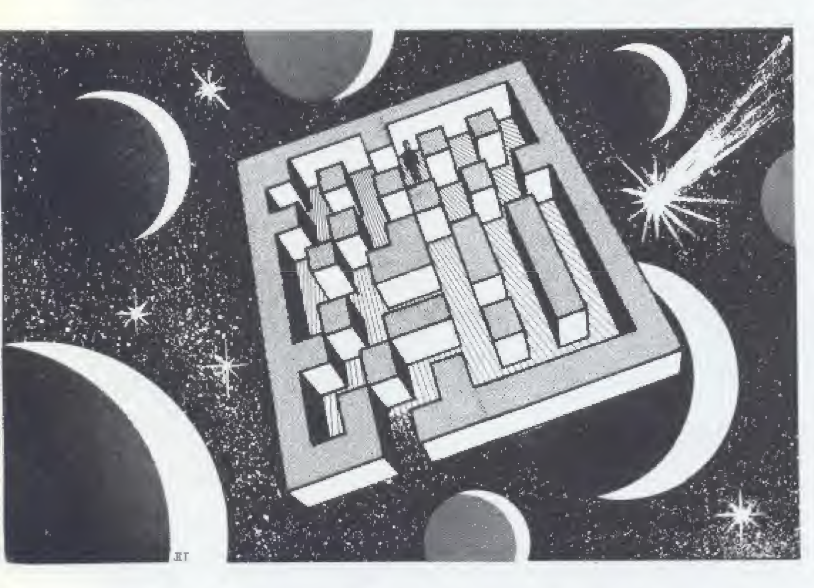


```
 158 IF Q<>8 THEN  GO TO 164
 160 LET S=S+1: IF S>4 THEN  LET
 S=S-4
 162 GO TO 100*S+115
 164 IF Q=6 THEN  GO TO 105+100*
S
 166 IF Q=7 THEN  GO TO 110+100*
S
 168 IF Q=0 THEN  GO SUB 700
 170 GO TO 115+100*S
 205 IF L(I+1,J)<>1 THEN  LET I=
I+1: GO TO 214
 207 GO SUB 650: GO TO 214
 210 IF L(I-1,J)<>1 THEN  LET I=
I-1: GO TO 214
 212 GO SUB 650
 214 GO SUB 680
 215 FOR T=1 TO 6
 220 IF L(I-T,J)<1 THEN  NEXT T
 222 IF L(I-1,J)=2 THEN  LET N=1
 225 FOR Y=0 TO T-1
 230 FOR Z=-1 TO 1 STEP 2
 235 LET A=1000+100*Y+10*(Z+1)
 240 IF L(I-Y,J+Z)=1 THEN  GO SU
B A: GO TO 250
 245 GO SUB A+10
 250 NEXT Z: NEXT Y
 255 GO TO 600
 305 IF L(I,J-1)<>1 THEN  LET J=
J-1: GO TO 314
 307 GO SUB 650: GO TO 314
 310 IF L(I,J+1)<>1 THEN  LET J=
J+1: GO TO 314
 312 GO SUB 650
 315 FOR T=1 TO 6
 320 IF L(I,T+J)<>1 THEN  NEXT T
 323 IF L(I-1,J+1)=2 THEN  LET M
=1
 325 FOR Y=0 TO T-1
 330 FOR Z=-1 TO 1 STEP 2
 335 LET A=1000+100*Y+10*(Z+1)
 340 IF L(I+Z,J+Y)>=1 THEN  GO S
UB A: GO TO 350
 345 GO SUB A+10
 350 NEXT Z: NEXT Y
 355 GO TO 600
 405 IF L(I-1,J)<>1 THEN  LET I=
I-1: GO TO 414
 407 GO SUB 650: GO TO 414
 410 IF L(I+1,J)<>1 THEN  LET I=
I+1: GO TO 414
 412 GO SUB 650
 414 GO SUB 680
 415 FOR T=1 TO 6
 420 IF L(I+T,J)<>1 THEN  NEXT T
 425 FOR Y=0 TO T-1
 430 FOR Z=1 TO -1 STEP -2
 435 LET A=1000+100*Y+10*(1-Z)
 440 IF L(I+Y,J+Z)=1 THEN  GO SU
B A: GO TO 450
 445 GO SUB A+10
 450 NEXT Z: NEXT Y
 455 GO TO 600
 505 IF L(I,J+1)<>1 THEN  LET J=
J+1: GO TO 514
```

```
 507 GO SUB 650: GO TO 514
 510 IF L(I,J-1)<>1 THEN  LET J=
J-1: GO TO 514
 512 GO SUB 650
 515 FOR T=1 TO 6
 520 IF L(I,J-T)<>1 THEN  NEXT T
 525 FOR Y=0 TO T-1
 530 FOR Z=1 TO -1 STEP -2
 535 LET A=1000+100*Y+10*(1-Z)
 540 IF L(I+Z,J-Y)>=1 THEN  GO S
UB A: GO TO 550
 545 GO SUB A+10
 550 NEXT Z: NEXT Y
 600 GO SUB 1600+10*T
 610 GO TO 150
 650 PRINT AT 20,9; FLASH 1;"NO
HAY PASO": RETURN
 680 IF L(I,J)=2 THEN  GO TO 180
0
 690 RETURN
 700 LET D=((ABS (I-1))^2+(ABS (
J-9))^2)^.5: PRINT AT 21,3;"DIST
.= ";INT (D*10+.5)/10: RETURN
1000 PLOT 0,0: DRAW 7,7: PLOT 0,
175: DRAW 7,-2: IF Y=T-1 THEN  D
RAW 0,-166
1001 RETURN
1010 PLOT 0,8: DRAW 7,0: IF Y<T-
1 THEN  DRAW 0,166
1011 PLOT 0,173: DRAW 7,0: RETUR
N
1020 PLOT 255,0: DRAW -7,7: PLOT
 255,175: DRAW -7,-2: IF Y=T-1 T
HEN  DRAW 0,-166
1021 RETURN
1030 PLOT 255,8: DRAW -7,0: IF Y
<T-1 THEN  DRAW 0,166
1031 PLOT 255,173: DRAW -7,0: RE
TURN
1100 PLOT 8,8: DRAW 55,55: PLOT
8,173: DRAW 55,-13: IF Y=T-1 THE
N  DRAW 0,-96
1101 IF M=1 AND S=2 THEN  PLOT 2
0,20: DRAW 0,136: DRAW 35,-4: DR
AW 0,-96
1102 LET M=0: RETURN
1110 PLOT 8,8: DRAW 0,166: PLOT
8,64: DRAW 55,0: IF Y<T-1 THEN
DRAW 0,96
1111 PLOT 8,159: DRAW 55,0: RETU
RN
1120 PLOT 247,8: DRAW -55,55: PL
OT 247,173: DRAW -55,-13: IF Y=T
-1 THEN  DRAW 0,-96
1121 RETURN
1130 PLOT 247,8: DRAW 0,166: PLO
T 247,64: DRAW -55,0: IF Y<T-1 T
HEN  DRAW 0,96
1131 PLOT 247,159: DRAW -55,0: R
ETURN
1200 PLOT 64,64: DRAW 30,30: PLO
T 64,159: DRAW 30,-7: IF Y=T-1 T
HEN  DRAW 0,-58
1201 RETURN
1210 PLOT 64,64: DRAW 0,96: PLOT
 64,94: DRAW 30,0: IF Y<T-1 THEN
  DRAW 0,58
1211 PLOT 64,152: DRAW 30,0: RET
```

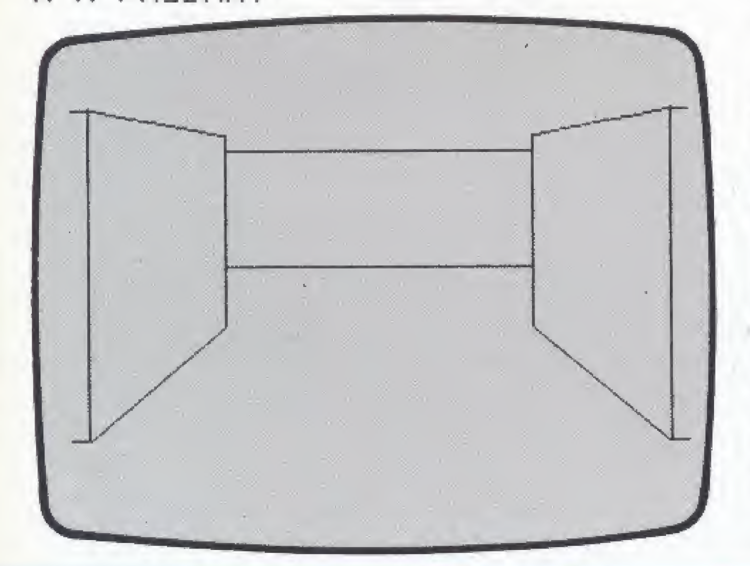


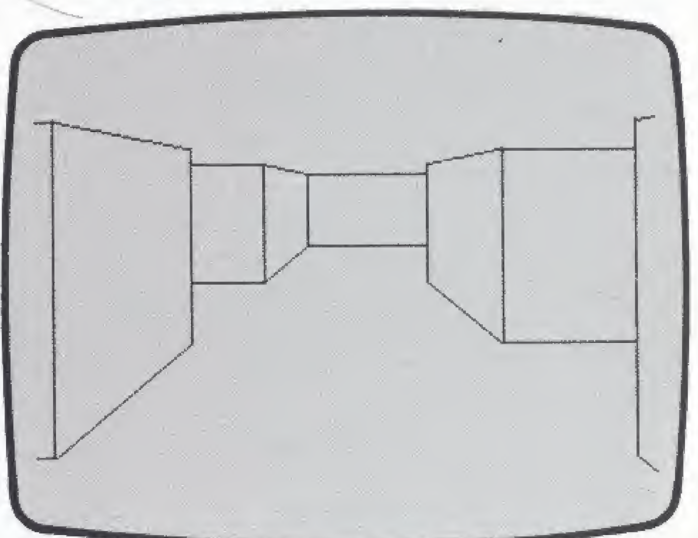
```
URN
1220 PLOT 191,64: DRAW -30,30: P
LOT 191,159: DRAW -30,-7: IF Y=T
-1 THEN  DRAW 0,-58
1221 RETURN
1230 PLOT 191,64: DRAW 0,96: PLO
T 191,94: DRAW -30,0: IF Y<T-1 T
HEN  DRAW 0,58
1231 PLOT 191,152: DRAW -30,0: R
ETURN
1300 PLOT 94,94: DRAW 18,18: PLO
T 94,152: DRAW 18,-5: IF Y=T-1 T
HEN  DRAW 0,-35
1301 RETURN
1310 PLOT 94,94: DRAW 0,58: PLOT
```



```
 94,112: DRAW 18,0: IF Y<T-1 THE
N  DRAW 0,35
1311 PLOT 94,147: DRAW 18,0: RET
URN
1320 PLOT 161,94: DRAW -18,18: P
LOT 161,152: DRAW -18,-5: IF Y=T
-1 THEN  DRAW 0,-35
1321 RETURN
1330 PLOT 161,94: DRAW 0,58: PLO
T 161,112: DRAW -18,0: IF Y<T-1
THEN  DRAW 0,35
1331 PLOT 161,147: DRAW -18,0: R
ETURN
1400 PLOT 112,112: DRAW 10,10: P
LOT 112,147: DRAW 10,-2: IF Y=T-
1 THEN  DRAW 0,-23
1401 LET V=1: RETURN
1410 PLOT 112,112: DRAW 0,35: PL
OT 112,122: DRAW 10,0: IF Y<T-1
THEN  DRAW 0,18
1411 PLOT 112,144: DRAW 10,0: RE
TURN
1420 PLOT 143,112: DRAW -10,10:
PLOT 143,147: DRAW -10,-2: IF Y=
T-1 THEN  DRAW 0,-23
1421 RETURN
1430 PLOT 143,112: DRAW 0,35: PL
OT 143,122: DRAW -10,0: IF Y<T-1
 THEN  DRAW 0,22
1431 PLOT 143,144: DRAW -10,0: R
ETURN
1500 PLOT 122,122: DRAW 6,6: DRA
W 0,15: DRAW -6,1: RETURN
1510 PLOT 122,127: DRAW 6,0: PLO
T 122,143: DRAW 6,0: PLOT 122,12
3: DRAW 0,22: RETURN
1520 PLOT 133,122: DRAW -6,6: DR
AW 0,15: DRAW 6,1: RETURN
1530 PLOT 133,127: DRAW -6,0: PL
OT 133,143: DRAW -6,0: PLOT 133,
123: DRAW 0,22: RETURN
1610 PLOT 8,8: DRAW 239,0: PLOT
8,173: DRAW 239,0
1613 IF N=1 THEN  LET N=0: PLOT
56,8: DRAW 0,152: DRAW 135,0: DR
AW 0,-152: PRINT AT 9,13; FLASH
1;"SALIDA"
1614 RETURN
1620 PLOT 64,64: DRAW 127,0: PLO
T 64,159: DRAW 127,0
1623 RETURN
1630 PLOT 94,94: DRAW 67,0: PLOT
 94,152: DRAW 67,0
1633 RETURN
1640 PLOT 112,112: DRAW 31,0: PL
OT 112,147: DRAW 31,0
1643 RETURN
1650 PLOT 122,122: DRAW 11,0: PL
OT 122,144: DRAW 11,0
1670 RETURN
1800 CLS : FOR I=1 TO 7
1810 PRINT AT 10,8; INK I;"YA HA
S SALIDO": BEEP .4,7*I: BEEP .2,
3*I
1820 NEXT I
```

```
1830 PRINT AT 13,10; FLASH 1;"EN
HORABUENA": PAUSE 150
1840 PRINT AT 15,5;"DESEAS VOLVE
R A PROBAR?"
1850 IF INKEY$="" THEN GO TO 1850
1860 IF INKEY$="S" OR INKEY$="s"
 THEN  CLS : GO TO 100
1870 STOP
```





Programas



Ediset

En bastantes ocasiones, a la hora de hacer un programa, hemos tenido que recurrir a crear una serie de caracteres, símbolos o dibujos que no se encontraban en el *set* usual de nuestro ordenador. Una ñ, una raya diagonal, o incluso un pequeño alienígena nos ha quitado alguna hora de sueño en algún momento dado de nuestra carrera como programadores. Tras un intenso visionado al manual de nuestro Spectrum y haciendo uso de algún programa, hemos conseguido lo que buscábamos. Para muchos ya es una rutina el tener que diseñar algún caracter especial.

Disponemos de 21 caracteres, llamados gráficos, libres para hacer con ellos toda clase de engendros y perversidades. Pero a veces este número es pequeño. Necesitamos más.

El programa que os proponemos a continuación podrá salvaros de este inconveniente en bastantes aspectos y situaciones. En primer lugar vamos a analizar línea por línea la forma en que está programado para una mejor comprensión de lo que sucede en las entrañas del ordenador cuando deis al RUN (tecla de muchas alegrías y decepciones).

Línea 1:

Colores del borde, papel y tinta. El OVER O es para asegurarnos este estado en todo el programa menos en las líneas que nos interese meterlo.

Línea 5:

Tres siniestros POKES. El primero hace que al pulsar cualquier tecla suene un corto pitido (variable del sistema PIP). Los otros dos corresponden a la variable CHARS que es la que determina en qué dirección se empieza a leer el conjunto de 256 caracteres que componen el set del Spectrum.

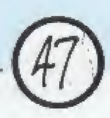
Línea 10:

Reservamos memoria a partir de la dirección 64600. Esto lo hacemos para que el programa en BASIC no se nos cuele en estas direcciones en caso de que sea excesivamente largo.

```
   1 BORDER 0: PAPER 0: INK 7:
OVER 0: CLS
   5 POKE 23609,75: POKE 23606,0
: POKE 23607,60
  10 CLEAR 64599
  12 FOR n=0 TO 50: PRINT  INK 1
+RND*6;"EDISET     ";: NEXT n: PR
INT AT 17,6; Jose Carlos Tomas
"
  15 LET di=1: LET c$=" ": DIM a
$(8,8): LET x1=0: LET s=64600: L
ET x=0: LET y=0
  16 FOR m=1 TO 8: FOR n=1 TO 8:
 LET a$(m,n)=STR$ 0: NEXT n: NEX
T m
  17 REM  COPIA ROM en RAM 64600

  20 FOR a=1 TO 767
  30 LET l=PEEK (15616+a)
  35 PRINT AT 19,7; OVER 1;"ESPE
RA 30 SEGUNDOS"
  40 POKE s+a,l
  50 NEXT a
  52 FOR n=1 TO 5: BEEP .1,45: N
EXT n
  55 GO SUB 300
  57 LET x2=0: LET y2=0
  60 GO TO 1000
 300 REM  PANTALLA
 305 CLS
 307 PRINT AT 0,0;"  ! ";"""";"
^ $ % & ' ( ) * + , - . /";AT 1,
0;"0 1 2 3 4 5 6 7 8 9 : ; < = >
 ?";AT 2,0;"@ A B C D E F G H I
J K L M N O";AT 3,0;"P Q R S T U
 V W X Y Z ¡ Ñ ¿ ^ _";AT 4,0;"#
a b c d e f g h i j k l m n o";A
T 5,0;"p q r s t u v w x y z ¨ ñ
 } ~";AT 6,0;"A B C D E F G H
 I J K L M N O P";AT 7,0;"Q R
 S T U"
 309 FOR n=10 TO 18: PRINT AT n,
2;"          ": NEXT n: PRINT AT
8,12;" ": FOR m=1 TO 8: FOR n=1
TO 8: LET a$(m,n)=STR$ 0: NEXT n
: NEXT m
 310 FOR n=16 TO 90 STEP 8: PLOT
 n,31: DRAW  BRIGHT 1;0,64: NEXT
 n
 315 FOR n=31 TO 96 STEP 8: PLOT
 16,n: DRAW  BRIGHT 1;64,0: NEXT
 n
 320 PLOT 8,23: DRAW  BRIGHT 1;8
0,0: DRAW  BRIGHT 1;0,80: DRAW
```

Guía del comprador de Todospectrum



Línea 12:

Carátula de presentación con la palabra EDISET con tinta aleatoria (INK 1+RND*6).

Línea 15:

Definimos diversas variables.

Línea 16:

Definimos por medio de dos bubles la matriz a$. STR$ convierte números en cadenas por lo tanto todos los valores de esta matriz son iguales a «O».

Línea 20 a 50:

Copiamos el juego de caracteres de la ROM a las posiciones de memoria de RAM 64600 en adelante. Leyendo con PEEK y metiéndolo en la RAM con POKE. La frase ESPERA 30 SEGUNDOS con OVER 1 hace que ésta parpadee de forma rápida sin ser un FLASH.

Línea 52:

Aviso de final de copia.

Línea 53:

Dos de los POKES siniestros cambian de valor. Ese nuevo valor resulta ser la nueva dirección donde el ordenador leerá el *set* de caracteres. ¡Exacto! Ahora todos los símbolos, letras y demás serán leídos de la RAM, la cual, nosotros, podemos modificar a nuestro placer.

Línea 55:

Saltamos a la subrutina 300.

Línea 57:

Definimos dos variables para el movimiento del cursor.

Línea 60:

Saltamos a la línea 1000.

Línea 300 a 330:

Subrutina para dibujar la pantalla principal de trabajo. En principio la línea 307 tenía unos bucles para «printar» en pantalla todos los caracteres (FOR n=32 TO 127:PRINT CHR$ n; "";:NEXT n) pero haciendolo así, tal como está, la impresión es mucho más rápida.

Línea 600 a 699:

Estas líneas son en realidad un programa aparte que permite dibujar en una rejilla, punto a punto, nuestros nuevos símbolos o modificar los ya existentes. Posee sus propias subrutinas de movimiento de cursor totalmente independiente de la parte principal del programa.

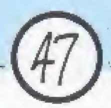
Línea 605:

Pitido de aviso de entrada en esta opción. Cambiamos de nuevo los famosos POKES para imprimir por pantalla los valores en decimal de la matriz a$ con la expresión

```
BRIGHT 1;-80,0: DRAW  BRIGHT 1;0
,-80
 325 PLOT 151,88: DRAW 24,0: DRA
W 0,24: DRAW -24,0: DRAW 0,-24
 327 POKE 23606,0: POKE 23607,60
: FOR n=1 TO 8: PRINT AT n+9,12;
VAL ("BIN "+a$(n,1 TO 8));"  ":
NEXT n: POKE 23606,88: POKE 2360
7,251
 330 RETURN
 600 REM  Dibuja en rejilla
 605 BEEP .1,30: POKE 23606,0: P
OKE 23607,60: FOR n=1 TO 8: PRIN
T AT n+9,12;VAL ("BIN "+a$(n,1 T
O 8)): NEXT n
 610 LET x=10: LET y=2
 620 PRINT AT 20,0;"cambiar , :v
olver:borra

                  "
 621 LET w1=10: LET w2=17: LET w
3=9: LET w4=2
 622 IF in OR es THEN  GO TO 650
 623 POKE 23606,88: POKE 23607,2
51
 625 IF di=1 THEN  PRINT AT 8,12
;c$
 650 LET e$=INKEY$
 651 PRINT  BRIGHT 1; OVER 1;AT
x,y;"■"
 652 IF e$=CHR$ 11 THEN  GO SUB
800
 654 IF e$=CHR$ 10 THEN  GO SUB
810
 656 IF e$=CHR$ 9 THEN  GO SUB 8
20
 658 IF e$=CHR$ 8 THEN  GO SUB 8
30
 665 IF e$="," THEN  PLOT  OVER
1;y+94,x+121-(x*2): PRINT  BRIGH
T 1; OVER 1;AT x,y;"■": BEEP .05
,30: GO SUB 700
 670 PRINT  BRIGHT 1; OVER 1;AT
x,y;"■"
 675 IF e$="v" THEN  LET di=0: G
O TO 1000
 680 IF e$="b" THEN  LET di=1: L
ET c$="  ": PRINT AT 8,12;c$: GO
SUB 309
 685 POKE 23606,0: POKE 23607,60
: PRINT AT x,12;VAL ("BIN "+a$(x
-9,1 TO 8));"  ": POKE 23606,88:
 POKE 23607,251
 699 GO TO 650
 700 IF a$(x-9,y-1)=STR$ 0 THEN
 LET a$(x-9,y-1)=STR$ 1: RETURN
 705 IF a$(x-9,y-1)=STR$ 1 THEN
 LET a$(x-9,y-1)=STR$ 0: RETURN
 800 IF x<=w1 THEN  RETURN
```

Programas



VAL ("BIN "+a$ (n,1 TO 8)). Este cambio de valores de los POKES que observaréis a lo largo del programa hace que se tomen los caracteres de la ROM o de la RAM según nos interese para tener con toda claridad en pantalla la información necesaria en caso de que hubiésemos cambiado o transformado algún número o letra por otra cosa.

Línea 621:

Estas variables tienen como misión establecer unos topes al movimiento del cursor por la pantalla, puesto que utilizamos siempre la misma subrutina. En la pantalla principal el cursor se mueve a todo lo largo de la pantalla y aquí solo en un pequeño cuadrado. Los valores w1 w2 w3 y w4 son alterados para este fin.

Línea 650 a 658:

Movimiento del cursor. Pensado para hacerlo con los cursores y no con los números, puesto que en principio es más cómodo para los poseedores del PLUS. Al final explicaremos los cambios que se pueden hacer.

Línea 665:

Cuando pulsamos la coma ocurren varias cosas. Dibujamos un punto en la posición superior derecha de la rejilla que es una reproducción exacta de cómo quedará nuestro diseño. Dibujamos un cuadrado blanco en la rejilla o panel de trabajo. Saltamos a la subrutina 700 que cambia el estado de una de las variables de la matriz a$. Si es cero la pone a uno y viceversa.

Línea 680:

Si pulsamos la B borramos la rejilla y la matriz a$ saltando a la línea 309 que forma parte de la subrutina que dibuja la pantalla principal. Siempre que queramos podemos saltar a la parte de una determinada subrutina que nos interese.

Línea 685:

Esta línea actualiza en todo momento en valor en decimal de cada línea del caracter a crear.

Línea 1001:

Programa principal. Definimos nuevas variables.

Línea 1003:

Siempre que veamos esta expre-

```
 805 PRINT  BRIGHT 1; OVER 1;AT
x,y;"■": LET x=x-1: PRINT  BRIGH
T 1; OVER 1;AT x,y;"■": RETURN
 810 IF x>=w2 THEN  RETURN
 815 PRINT  BRIGHT 1; OVER 1;AT
x,y;"■": LET x=x+1: PRINT  BRIGH
T 1; OVER 1;AT x,y;"■": RETURN
 820 IF y>=w3 THEN  RETURN
 825 PRINT  BRIGHT 1; OVER 1;AT
x,y;"■": LET y=y+1: PRINT  BRIGH
T 1; OVER 1;AT x,y;"■": RETURN
 830 IF y<=w4 THEN  RETURN
 835 PRINT  BRIGHT 1; OVER 1;AT
x,y;"■": LET y=y-1: PRINT  BRIGH
T 1; OVER 1;AT x,y;"■": RETURN
1000 REM Eleccion de caracter
1001 POKE 23606,0: POKE 23607,60
: LET in=0: LET es=0: LET g=1
1002 PRINT AT 15,20;"Impresora";
AT 16,20;"1 ESPEJO";AT 17,20;"2
INVERTIR";AT 18,20;"3 INVERSO";A
T 14,20;"Run";AT 20,0;"posiciona
r:dibujar:editar:borra  leer:10a
d: Graba:reponer:Salir": POKE 23
606,88: POKE 23607,251
1003 LET e$=INKEY$
1004 IF e$="3" THEN  BEEP .01,29
: GO SUB 1150
1005 PRINT  OVER 1;AT x2,y2;"■"
1006 IF e$="2" THEN  LET di=1: L
ET c$=CHR$ w1: BEEP .01,29: GO S
UB 1160: GO SUB 309: GO SUB 1200
: GO SUB 327: GO SUB 1300: BEEP
.5,20: GO SUB 1160: PRINT AT x2,
y2; OVER 1;"■"
1007 IF e$="1" THEN  LET di=1: L
ET c$=CHR$ w1: BEEP .01,29: GO S
UB 1170: GO SUB 309: GO SUB 1200
: GO SUB 327: GO SUB 1300: BEEP
.5,20: GO SUB 1170: PRINT AT x2,
y2; OVER 1;"■"
1008 IF e$="R" THEN  RUN
1009 IF e$="r" THEN  GO TO 53
1010 IF e$="p" THEN  PRINT  OVER
 1;AT x2,y2;"■": GO TO 2000
1013 IF e$=CHR$ 8 THEN  GO SUB 1
075
1015 IF e$=CHR$ 9 THEN  GO SUB 1
085
1020 IF e$=CHR$ 11 THEN  GO SUB
1095
1023 IF e$=CHR$ 10 THEN  GO SUB
1105
1024 IF e$="S" THEN  CLS : POKE
23606,0: POKE 23607,60: PRINT AT
 11,1;"EDISET   GOTO 53 Continua
r              RUN      Empezar"
: STOP
1025 IF e$="G" THEN  GO SUB 1400
1026 IF e$="O" THEN  GO SUB 1500
1027 IF e$="1" THEN  LET di=1: P
RINT AT 8,12;" ": GO SUB 309: GO
```

sión (LET e$=INKEY$) convertimos la variable e$, u otra cualquiera, en la misma función que desempeña INKEY$: leer el teclado y comprobar si hay alguna tecla pulsada.

Línea 1004:

Al pulsar la tecla 3 el programa salta a la subrutina 1150 que es la encargada de invertir tinta por papel y viceversa.

Línea 1005:

Esta línea, junto con la 1030, hace que el cursor parpadee e impide que borre la pantalla por donde pasa.

Línea 1006:

Al pulsar el 2 el programa salta a una serie de subrutinas y tenemos como resultado el caracter elegido pero boca abajo.

Línea 1007:

Al pulsar el 1 tenemos la imagen espejo del caracter elegido.

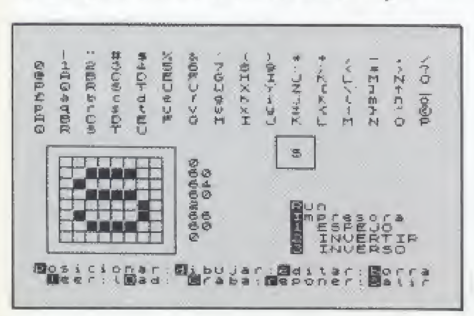
Línea 1008:

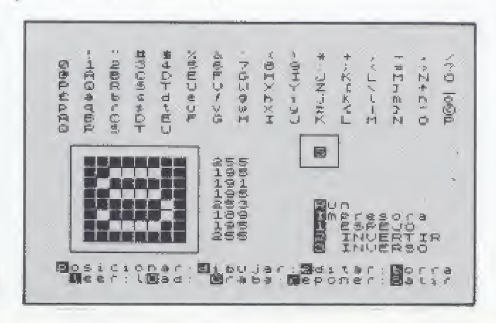
Pulsamos la R mayúscula y hacemos un RUN lo cual empezamos de nuevo borrando todo lo que hay en la RAM. Sólo permanecen inalterables los UGD.

Línea 1009:

Al pulsar la «r» minúscula saltamos a la línea 53, dibujando de nuevo la pantalla principal y devolviendo el cursor a la parte superior izquierda. La utilidad de esta instrucción, aparentemente inútil, está pensada para el caso siguiente: Imaginemos que editamos en el lu-

```
 SUB 1200: GO SUB 327: BEEP .1,4
5: LET c$=CHR$ w1
1028 IF e$="b" THEN  PRINT AT 8,
12;" ": LET di=1: LET c$=" ": GO
 SUB 309
1029 IF e$="e" THEN  GO SUB 1300
: GO TO 1003
1030 PRINT  OVER 1;AT x2,y2;"█"
1031 LET w=INT ((x1+y2)/2): LET
w1=w+32
1032 IF g=1 THEN  PRINT  BRIGHT
1;AT 9,20;CHR$ w1
1033 IF e$="d" THEN  GO TO 600
1034 IF g=0 THEN  PRINT  BRIGHT
1; INVERSE 1;AT 9,20;CHR$ w1
1035 IF x2<=5 THEN  LET x1=x2*32
1036 IF x2>5 THEN  LET x1=32+(x2
*32)
1045 IF e$="i" THEN  LET di=1: P
RINT AT 8,12;" ": GO SUB 309: GO
 SUB 1200: GO SUB 327: LET c$=CH
R$ w1: POKE 23606,0: POKE 23607,
```





El corcho

gar destinado al SPACE un dibujo cualquiera. Cuando salgamos de la pantalla principal y entremos en ella otra vez observaremos que todos los espacios son rellenados con nuestro dibujo, lo cual produce un molesto visionado de la pantalla. Si lo hemos hecho «aposta», nos tendremos que aguantar con las consecuencias pero si ha sido un error podemos editar un «vacío» de nuevo y apretando la «n» todo volverá a la normalidad. Haced el experimento.

Línea 1010:

Pulsamos la «P» y saltamos a un subprograma cuya única misión es la de poder editar en pantalla los resultados finales dibujando en ella a nuestro antojo y por medio de un cursor todo lo que hubiésemos creado.

Línea 1024:

Pulsamos la «s» mayúscula y salimos del programa.

Línea 1025:

Pulsando la «g» mayúscula pasamos a la subrutina de grabación.

Línea 1026:

Pulsando la «o» mayúscula pasamos a la subrutina de carga de un *set* almacenado con anterioridad en cinta.

Línea 1027:

Pulsando la «l» pasamos a la rejilla el caracter elegido con el cursor. Este caracter podrá ser modificado y editado de nuevo en la misma u otra posición.

Línea 1028:

Pulsando la «b» borramos la rejilla.

```
60: GO SUB 1600: POKE 23606,88:
POKE 23607,251
1070 GO TO 1003
1075 IF y2<=0 THEN  RETURN
1080 PRINT AT x2,y2; OVER 1;"█":
 LET y2=y2-2: PRINT  OVER 1;AT x
2,y2;"█": RETURN
1085 IF y2>=30 THEN  RETURN
1086 IF y2>=7 AND x2=7 THEN  RET
URN
1090 PRINT AT x2,y2; OVER 1;"█":
 LET y2=y2+2: PRINT  OVER 1;AT x
2,y2;"█": RETURN
1095 IF x2<=0 THEN  RETURN
1100 PRINT AT x2,y2; OVER 1;"█":
 LET x2=x2-1: PRINT  OVER 1;AT x
2,y2;"█": RETURN
1105 IF x2>=7 THEN  RETURN
1106 IF x2>=6 AND y2>=9 THEN  RE
TURN
1110 PRINT AT x2,y2; OVER 1;"█":
 LET x2=x2+1: PRINT  OVER 1;AT x
2,y2;"█": RETURN
1150 IF g=0 THEN  LET g=1: PRINT
  OVER 1;AT 18,20;"          ": R
ETURN
1152 IF g=1 THEN  LET g=0: PRINT
  PAPER 2; OVER 1; BRIGHT 1;AT 1
8,20;"          ": RETURN
1160 IF in=0 THEN  LET in=1: PRI
NT  PAPER 2; OVER 1; BRIGHT 1;AT
 17,20;"          ": RETURN
1162 IF in=1 THEN  LET in=0: PRI
NT  OVER 1;AT 17,20;"          "
: RETURN
1170 IF es=0 THEN  LET es=1: PRI
NT  PAPER 2; OVER 1; BRIGHT 1;AT
 16,20;"          ": RETURN
1172 IF es=1 THEN  LET es=0: PRI
NT  OVER 1;AT 16,20;"          ":
RETURN
1200 REM Dibuja carac en reja
1205 LET a1=10: LET b1=2
1208 IF in=0 THEN  FOR m=103 TO
95 STEP -1: LET b1=2
1210 IF in=1 THEN  FOR m=96 TO 1
05: LET b1=2
1213 IF es=1 THEN  FOR n=167 TO
159 STEP -1
1214 IF es=0 THEN  FOR n=160 TO
168
1215 IF POINT (n,m)=1 THEN  PRIN
T  BRIGHT 1; OVER 1;AT a1,b1;"█"
: LET a$(a1-9,b1-1)=STR$ 1
1220 LET b1=b1+1
1250 NEXT n: LET a1=a1+1: NEXT m
: RETURN
1300 REM  EDITAR
1315 FOR n=1 TO 8: LET b$=a$(n,1
)+a$(n,2)+a$(n,3)+a$(n,4)+a$(n,5
)+a$(n,6)+a$(n,7)+a$(n,8)
1317 IF w1<144 THEN  POKE s+(w*8
)+n-1,VAL ("BIN "+b$): NEXT n
1319 IF w1>=144 THEN  POKE USR "
a"+((w-112)*8)+n-1,VAL ("BIN "+b
$): NEXT n
1322 PRINT AT x2,y2;CHR$ w1
1325 BEEP .1,35: RETURN
1400 REM  grabar en casette
1405 POKE 23606,0: POKE 23607,60
1410 CLS : PRINT AT 3,10;"SAVE";
AT 5,10;"1  SET";AT 7,10;"2  UGD
```

```
";AT 9,10;"3  Ambos"
1412 INPUT "Nombre fichero:";c$:
 IF c$="" OR LEN c$>10 THEN  GO
TO 1412
1414 PRINT : PRINT TAB 5;"Pulsa
una opcion": PRINT : PRINT "Para
 utilizar el nuevo set en tuprog
rama incluye estos POKES:"'': PR
INT "POKE 23606,88 y POKE 23607,
251"'': PRINT "y CLEAR 64599 en
opciones 1 y 3"
1415 LET e$=INKEY$
1417 IF e$="1" THEN  CLS : PRINT
 "Opcion 1  SET"'': PRINT "Direc
cion de memoria 64500"''"Longitu
d 768": SAVE c$CODE s,768: GO SU
B 1450: GO TO 53
1419 IF e$="2" THEN  CLS : PRINT
 "Opcion 2  UDG"'': PRINT "Direc
cion de memoria USR ""a"""''"Lon
gitud 168": SAVE c$CODE USR "a",
168: GO SUB 1450: GO TO 53
1420 IF e$="3" THEN  CLS : PRINT
 "Opcion 3  SET y UDG"'': PRINT
"Direccion de memoria 64500"''"L
ongitud 936": SAVE c$CODE s,936:
 GO SUB 1450: GO TO 53
1430 GO TO 1415
1450 CLS : PRINT AT 19,0;"O.K. "
;c$;AT 21,0;"Quieres verificar?
(S/N)"
1452 IF INKEY$="s" THEN  CLS : P
RINT  INK 2; PAPER 7;AT 11,0;"GO
TO 53 en caso de error o BREAK":
 VERIFY ""CODE : CLS : PRINT AT
21,0;c$;" verificado": BEEP .1,4
0: PAUSE 100: RETURN
1455 IF INKEY$="n" THEN  RETURN
1457 GO TO 1452
1500 REM  Cargar set de cinta
1505 POKE 23606,0: POKE 23607,60
1513 CLS : PRINT AT 3,10;"LOAD";
AT 5,10;"1  SET";AT 7,10;"2  UGD
";AT 9,10;"3  Ambos"
1514 INPUT "Nombre fichero:";c$:
 PRINT AT 21,0;"Pulsa una opcion
"
1515 LET e$=INKEY$
1520 IF e$="1" THEN  LOAD c$CODE
 : GO TO 53
1525 IF e$="2" THEN  LOAD c$CODE
 USR "a": GO TO 53
1530 IF e$="3" THEN  LOAD c$CODE
 : GO TO 53
1535 GO TO 1515
1600 IF w1>=144 THEN  LPRINT  IN
VERSE 1;CHR$ (w1-79); INVERSE 0;
" ";CHR$ w1;" ";
1605 IF w1<144 THEN  LPRINT CHR$
 w1;" ";: POKE 23606,88: POKE 23
607,251: LPRINT CHR$ w1;" ";: PO
KE 23606,0: POKE 23607,60
1610 FOR n=1 TO 8: LPRINT VAL ("
BIN "+a$(n,1 TO 8));",";: NEXT n
: LPRINT '': RETURN
2000 BRIGHT 1: PAPER 6: BORDER 7
: INK 0: CLS : LET di=0: LET gr=
0: LET x=10: LET y=2: LET w1=0:
LET w2=18: LET w3=11: LET w4=0:
CLS
2001 POKE 23606,0: POKE 23607,60
2002 PRINT AT 19,21;"AT Copy";AT
 20,0;"SPACE borrar       ENTER g
raficosDELETE volver       EDIT m
odo E   "
2004 PRINT  INK 4;AT 0,13;" !";"
""";"^$%&' ()*+,-./0"
2005 PRINT  INK 4;AT 2,13;"12345
6789:;<=>?@A";AT 4,13;"BCDEFGHIJ
KLMNOPQR";AT 6,13;"STUVWXYZ¡ñ¿^_
#abc";AT 8,13;"defghijklmnopqrst
```



```
1610 FOR n=1TO 8:LPRINT VAL ("BI
N "+a$(n,1TO 8));",";:NEXT n:LPR
INT '':RETURN
2000 BRIGHT 1: PAPER 6: BORDER 7
: INK 0: CLS : LET di=0: LET gr=
0: LET x=10: LET y=2: LET w1=0:
LET w2=18: LET w3=11: LET w4=0:
CLS
2001 POKE 23606,0: POKE 23607,60
2002 PRINT AT 19,21;"AT Copy";AT
 20,0;"SPACE borrar       ENTER g
raficosDELETE volver       EDIT m
odo E   "
2004 PRINT  INK 4;AT 0,13;" !";"
""";"^$%&' ()*+,-./0"
2005 PRINT  INK 4;AT 2,13;"12345
6789:;<=>?@A";AT 4,13;"BCDEFGHIJ
KLMNOPQR";AT 6,13;"STUVWXYZ¡ñ¿^_
#abc";AT 8,13;"defghijklmnopqrst
```



```
";AT 10,13;"uvwxyz¨ñ} ABCDE";A
```

Línea 1029:
Pulsando la «e» pasamos lo que hay en la rejilla a la posición del cursor.

Línea 1031:
Se definen unas variables relacionadas con la posición del cursor y la posición de memoria RAM donde deberá ser almacenada.

Línea 1033:
Pulsando la «d» saltamos a la subrutina de dibujar en rejilla.

```
T 12,13;"FGHIJKLMNOPQRSTU"
2006 POKE 23606,88: POKE 23607,2
51
2007 PRINT  INK 1; BRIGHT 1;AT 1
,13;" !";"""";"^$%&'()*+,-./0"
2008 PRINT  INK 1; BRIGHT 1;AT 3
,13;"123456789:;<=>?@A";AT 5,13;
"BCDEFGHIJKLMNOPQR";AT 7,13;"STU
VWXYZ¡Ñ¿^_#abc";AT 9,13;"defghij
klmnopqrst";AT 11,13;"uvwxyz¨ñ}~
 ABCDE";AT 13,13;"FGHIJKLMNO
PQRSTU"
2009 POKE 23606,88: POKE 23607,2
51: PRINT  OVER 1; BRIGHT 1;AT x
,y;"[redacted]"
2010 LET e$=INKEY$
2011 IF e$="AT " THEN  COPY : GO
 TO 2010
2013 IF e$=CHR$ 8 THEN  GO SUB 8
30: GO TO 2010
2015 IF e$=CHR$ 9 THEN  GO SUB 8
20: GO TO 2010
2020 IF e$=CHR$ 11 THEN  GO SUB
800: GO TO 2010
2023 IF e$=CHR$ 10 THEN  GO SUB
810: GO TO 2010
2025 IF e$=CHR$ 13 THEN  BEEP .0
5,30: PRINT AT 20,24; OVER 1;"[redacted]";AT x,y; BRIGHT 1;" ": GO
 SUB 2200: BEEP .1,35: GO TO 201
0
2027 IF (e$>CHR$ 117 OR e$<CHR$
97) AND gr=47 THEN  GO TO 2010
2030 IF e$<>"" THEN  PRINT  BRIG
HT 1; INVERSE 1;AT x,y;CHR$ VAL
"(CODE e$)+gr"
2035 IF e$=CHR$ 12 THEN  PAPER 0
: INK 7: BRIGHT 0: BORDER 0: CLS
 : GO TO 53
2040 IF e$=CHR$ 7 THEN  PRINT  O
VER 1;AT 21,24;"[redacted]": BEEP .1
,20: GO SUB 2300
2100 GO TO 2010
2200 IF gr=0 THEN  LET gr=47: PR
INT AT 20,0;"                   ";AT
 21,0;"
      ": RETURN
2210 IF gr=47 THEN  LET gr=0: PO
KE 23606,0: POKE 23607,60: PRINT
 AT 20,0;"SPACE borrar";AT 21,0;
"DELETE volver      EDIT modo E"
: POKE 23606,88: POKE 23607,251:
 RETURN
2300 PRINT  BRIGHT 1;AT x,y;"[redacted]"
2310 PRINT AT 20,0;"
                              "
2320 LET e$=INKEY$
2321 IF e$=CHR$ 8 THEN  GO SUB 8
30: GO TO 2320
2322 IF e$=CHR$ 9 THEN  GO SUB 8
20: GO TO 2320
2323 IF e$=CHR$ 11 THEN  GO SUB
800: GO TO 2320
2324 IF e$=CHR$ 10 THEN  GO SUB
810: GO TO 2320
2325 IF e$="y" THEN  PRINT  INVE
RSE 1; BRIGHT 1;AT x,y;"¡"
2326 IF e$="u" THEN  PRINT  INVE
RSE 1; BRIGHT 1;AT x,y;"¿"
2327 IF e$="a" THEN  PRINT  INVE
RSE 1; BRIGHT 1;AT x,y;"~"
2328 IF e$="s" THEN  PRINT  INVE
RSE 1; BRIGHT 1;AT x,y;"ñ"
2329 IF e$="d" THEN  PRINT  INVE
RSE 1; BRIGHT 1;AT x,y;"Ñ"
2330 IF e$="f" THEN  PRINT  INVE
RSE 1; BRIGHT 1;AT x,y;"¨"
2331 IF e$="g" THEN  PRINT  INVE
RSE 1; BRIGHT 1;AT x,y;"}"
2332 IF e$="p" THEN  PRINT  INVE
RSE 1; BRIGHT 1;AT x,y;"
2335 IF e$=CHR$ 7 THEN  PRINT AT
 21,24; OVER 1;"[redacted]": BEEP .1
,35: POKE 23606,0: POKE 23607,60
: PRINT AT 20,0;"SPACE borrar
  ENTER graficos": POKE 23606,8
8: POKE 23607,251: RETURN
2337 IF e$=CHR$ 12 THEN  RETURN
2340 GO TO 2320
9999 SAVE "ediset" LINE 1: VERIF
Y ""
```

Línea 1045:

Pulsando la «i» podemos pasar a impresora los valores en decimal del caracter elegido con el cursor.

Línea 1075 a 1110:

Subrutinas de movimiento del cursor de la pantalla principal.

Línea 1150 y 1152:

Subrutina de inversión de papel por tinta y viceversa.

Línea 1160 y 1162:

Subrutina para invertir el caracter.

Línea 1170 y 1172:

Subrutina para crear una imagen espejo del caracter elegido.

Línea 1200 a 1250:

Dibuja en la rejilla el caracter elegido escogiendo unos bucles u otros dependiendo de ciertas variables elegidas de antemano en las subrutinas 1160 y 1170.

Línea 1300 a 1325:

Edita el caracter que hay en rejilla llevándolo a la posición del cursor.

Línea 1400 a 1457:

Subrutina de grabación.

Línea 1500 a 1535:

Subrutina de carga del cassette a ordenador de cualquier set almacenado en cinta con anterioridad.

Línea 2000 a 2340:

Subrutina que permite un visionado en conjunto de todos los caracteres pudiéndolos posicionar juntos o por separado dentro de unos márgenes.

Línea 9990:

Graba y verifica el propio programa.

Hasta aquí la explicación un poco exhaustiva del programa. Pero antes de pasar a explicar cómo utilizarlos hay algunas cosas que se deben explicar.

El programa está pensado para el PLUS en cuanto a manejo del teclado. Si tenéis el normal cambiad todos los IF e$=CHR$ 11 a 8 por IF e$="(letra que queráis)" teniendo en cuenta que el CHR$ 11 equivale al cursor arriba, el 10 abajo, el 9 derecha y el 8 izquierda. También podéis cambiar en la línea 665 IF e$="," por otra letra.

El programa permite crear los 21 gráficos definibles por el usuario (UDG) y además cambiar, modificar y alterar todo el *set* de caracteres, grabarlo en cinta para una utilización posterior y sacar por impresora algunos datos útiles.

Para entenderlo mejor lo ideal será teclear el programa, grabarlo en cinta, verificarlo y hacer algunas pruebas.

Una vez hecho esto le damos al RUN y si todo es correcto aparecerá la carátula y tendremos que esperar aproximadamente 30 segundos. Al cabo de este tiempo, se dibujará la pantalla principal y ya podemos empezar a trabajar.

Para empezar podemos crear la letra «ñ». Llevamos el cursor hasta la «n» y pulsamos la «l» de leer. El menú, como podréis observar, se encuentra en la parte inferior de la pantalla. En la rejilla se dibujará, en grande, la letra «n». Ahora pulsar la «d» y un cursor parpadeante aparecerá dentro de la rejilla desapareciendo al mismo tiempo el otro cursor y cambiando el menú en la línea inferior. Tenéis unos números a la derecha de la rejilla que son el valor en decimal de cada fila del caracter elegido y además una «n» encima de éstos. Llevar en cursor hacia la derecha 2 espacios y pulsar la coma. Sonará un BEEP Y el número de la derecha habrá cambiado a 32. Correr otro espacio y pulsar de nuevo la coma. El número cambia a 48 y con otro espacio más será de 56. Si observáis encima de esta columna de números lo que antes era una «n» se ha convertido en «ñ». Pulsar la «v» para volver al menú principal. Desaparecerá el cursor de la rejilla y aparecerá donde lo dejamos anteriormente. Ahora podemos desplazar este cursor donde nos interese colocar la «ñ». Por ejemplo, a la «N» que hay dos filas más abajo. Las dos últimas filas de letras mayúsculas corresponden a los UDG. Ahora pulsamos la «e». El cursor deja de parpadear por un momento y ya está. Tenemos una flamante «ñ» en el UDG«N».

José C. Tomás

DIRECTOR:
Simeón Cruz
COORDINADOR EDITORIAL:
Emiliano Juárez
REDACCION:
Juan Arencibia, Fernando García, José C. Tomás, Luis M. Brugarolas, Ricardo García, Santiago Gala
DISEÑO: Ricardo Segura

Editado por
PUBLINFORMATICA, S. A.
Presidente: Fernando Bolín
Director Editorial: Norberto Gallego

Administración:
INFODIS, S. A.
Gerente de Circulación y ventas:
Luis Carrero
Producción:
Miguel Onieva
Director de Marketing:
Antonio González
Servicio al cliente:
Julia González. Tel. 733 79 69
Administración:
Miguel Atance y Antonio Torres
Jefe de Publicidad
María José Martín
Dirección y redacción:
Bravo Murillo, 377-5.º A. Tel. 733 74 13
Telex: 48877 OPZX e 28020 Madrid
Administración y Publicidad:
Bravo Murillo, 377-3 E. Tels. 733 96 62/96
Publicidad Madrid:
María Jose Martín
Publicidad Barcelona:
María del Carmen Ríos, Olga Martorell. Pelayo, 12. Tel. (93) 318 02 89. 08001 Barcelona.
Depósito legal: M-29041-1984
Distribuye S.G.E.L.
Avda. Valdelaparra, s/n. Alcobendas-Madrid.
Fotomecánica: Karmat, C/ Pantoja, 10. Madrid.
Fotocomposición: Artecomp.
Imprime: Héroes, C/ Torrelara, 8. Madrid.
Esta publicación es miembro de la Asociación de Revistas de Información ARI asociada a la Federación Internacional de Prensa Periódica, FIPP.

SUSCRIPCIONES:
Rogamos dirijan toda la correspondencia relacionada con suscripciones a:
TODOSPECTRUM
EDISA: Tel. 415 97 12
C/ López de Hoyos, 141-5.º
28002 MADRID
(Para todos los pagos reseñar solamente TODOSPECTRUM)
Para la compra de ejemplares atrasados diríjanse a la propia editorial
TODOSPECTRUM
C/ Bravo Murillo, 377-5.º A
Tel. 733 74 13 - 28020 MADRID

Si deseas colaborar en TODOSPECTRUM remite tus artículos o programas a Bravo Murillo 377, 5.º A. 28020 Madrid. Los programas deberán estar grabados en cassette y los artículos mecanografiados.

A efectos de remuneración, se analiza cada colaboración aisladamente, estudiando su complejidad y calidad.

SEIKOSHA SP-800

El fruto de la Investigación

La nueva impresora de **SEIKOSHA SP-800**, con un ordenador personal puede escribir ***96 combinaciones de letra diferentes***, desde 96 caracteres por segundo a 20 con muy alta calidad de letra, además es gráfica en alta densidad.

Su precio es de 69.900 Pts con ***introductor automático hoja a hoja.***

Con un pequeño ordenador personal, un procesador de textos puede costar alrededor de cien mil pesetas.

Infórmese y comprenderá por qué **las máquinas de escribir** tienen demasiados años.

Nuestra calidad es "SEIKO";
nuestros precios, únicos
Si desea más información,
consulte con nuestro distribuidor
más cercano, llame o escriba a:



DIRECCION COMERCIAL:
Av. Blasco Ibañez, 114-116
46022 VALENCIA
Tel.(96) 372 88 89
Télex 62220
DIRECCION COMERCIAL EN CATALUÑA:
C/Muntaner, 60-2-4Pta
08011 BARCELONA
Tel.(93) 323 32 19

ESTOS SON NUESTROS MODELOS:

MODELO		VELOCIDAD	COLUMNAS	TIPOS DE LETRA	P.V.P.R * INTERFACE PARALELO
GP-50	LA PEQUEÑA	40 cps	46	2	25.900
GP-500	LA ECONOMICA	50 "	80	2	47.900
GP-550	LA STANDARD	86 "	80-136	10	59.900
SP-800	LA PERFECCION	96 "	80-137	20	69.900
GP-700	LA DE COLOR	50 "	80-106	3	84.900
BP-5200	LA DE OFICINA	200 "	136-272	10	199.900
BP-5420	LA MAS RAPIDA	420 "	136-272	10	299.900

* Los precios indicados son los recomendados para conexión tipo paralelo Centronics, para otro tipo de conexión, sufren un ligero incremento.

Este pie de página ha sido realizado integramente con la nueva impresora:

SEIKOSHA SP-800